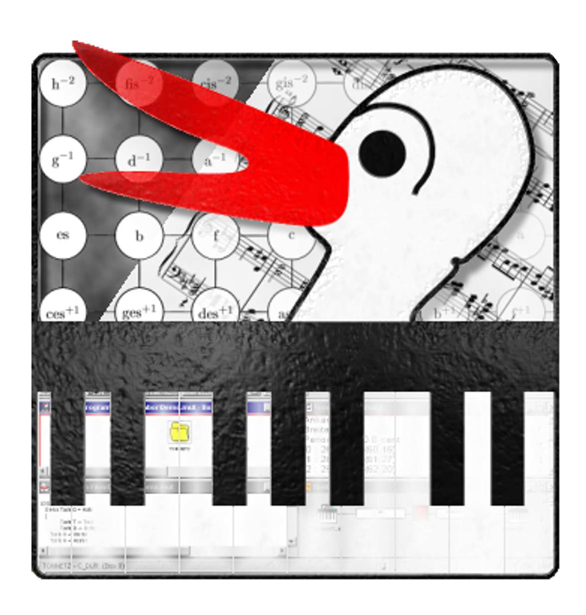
Programmier- und Bedienungshandbuch

Mutabor

Ein computergesteuertes Musikinstrument zum Experimentieren mit Stimmungslogiken und Mikrotönen

Volker Abel, Peter Reiss,
Rüdiger Krauße und Tobias Schlemmer

Programmversion 3.0x (2006)



Inhaltsverzeichnis

Vorwort

Das Projekt MUTABOR II wurde im August 1987 gegründet. Das ursprüngliche Ziel war es, das Instrument Mutabor, welches im Jahre 1984 im Rahmen des Forschungsvorhabens „Mathematische Musiktheorie“ an der TH Darmstadt gebaut wurde, auf einen handelsüblichen Rechner zu übertragen, da der Prototyp von Mutabor ein Unikat ist.

Aus diesem zunächst nur als kosmetische Korrektur geplanten Projekt entstand im Laufe der Zeit ein völlig neues und umfassenderes Konzept eines Instrumentes zum Experimentieren mit Stimmungslogiken und Mikrotönen. MUTABOR II wurde erstmals im Mai 1991 auf dem vierten internationalen Symposium für Mikrotonforschung im Mozarteum Salzburg vorgestellt.

MUTABOR 3 ist ein darauf aufbauendes Projekt, welches eine breitere Palette an technischen Möglichkeiten bereitstellt. MUTABOR 3 ist ein Projekt der TU Dresden.

Mutabor – Einerseits lateinisch: Ich werde verändert werden
Mutabor – andererseits **Mut**ierende **a**utomatisch **b**etriebene **Or**gel
Mutabor – oder das Zauberwort aus Kalif Storch
Mutabor – aber auch Mut, ab dem Ohr seltsame Dinge zu hören
Mutabor – und Mut, dass das Ohr abfallen könnte
Mutabor – . . .

Auf diesem Wege danken die Autoren allen, die ihnen bei der Entwicklung dieses Programmes so hilfreich zur Seite gestanden haben. Explizit genannt seien Herr Levigion, Herr Dr. Pense und Herr Dr. Schmitt, Uni Mainz, die zum konzeptuellen Entwurf viele Ideen eingebracht haben, Herrn Prof. Ganter und Herrn Prof. Wille, TH Darmstadt, deren Projekt „Mutabor“ hier seine Weiterentwicklung gefunden hat.

„Feinste Tonunterschiede werden mit dem computergesteuerten Musikinstrument MUTABOR II hörbar.“ (Darmstädter Echo)

„Für Experimentalmusiker bietet das computergestützte Gerät [. . .] einen schier unerschöpflichen Fundus für eigene Kreationen. [. . .] Das Instrument macht feinste Tonabfolgen in die Höhe oder Tiefe, über die man bisher nur theoretisch fachsimpeln konnte, endlich hörbar.“ (Frankfurter Rundschau)

Einschränkung der Gewährleistung

Inhaltliche Änderungen des Handbuchs und der Softwareprogramme behalten wir uns ohne Ankündigung vor. Es wird keine Haftung für die Richtigkeit des Inhaltes dieses Handbuchs oder Schäden, die sich aus dem Gebrauch der Softwareprogramme ergeben, übernommen. Ebenso können wir die Funktionsfähigkeit der Software nicht garantieren – *da sich Fehler, trotz aller Bemühungen, nie vollständig vermeiden lassen, sind wir für Hinweise jederzeit dankbar.*

Warenzeichen

Innerhalb dieses Handbuchs wird auf Warenzeichen Bezug genommen, die nicht explizit als solche ausgewiesen sind. Aus dem Fehlen einer Kennzeichnung kann also nicht geschlossen werden, dass ein Name frei von Rechten Dritter ist.

Teil I.

Einleitung

1. Das Konzept von Mutabor

Es gibt eine Fülle verschiedener Konzepte mikrotonaler Kompositionstechnik. Nahezu jeder Komponist hat ein eigenes Tonsystem und eigene Intonationsvorschriften. Die Tatsache, dass bis heute noch keine standardisierte Notation für Mikrotöne von allen Komponisten und Musikwissenschaftlern akzeptiert wird, lässt auf die ausgesprochene Vielfalt auf diesem Gebiet schließen.

Bei der Planung von MUTABOR wurde versucht, möglichst vielen Komponisten, Instrumentalisten und Musikwissenschaftlern ein Instrument in die Hand zu geben, das sich durch größtmögliche Flexibilität auszeichnet. Das Konzept, mit dem man dies realisieren kann, ist die Entwicklung einer möglichst universellen Programmiersprache zum Programmieren von Tonsystemen und Stimmungslogiken. Der Anwender kann sich die Stimmung seines Instrumentes ohne großen Aufwand individuell programmieren. Dies erspart den Bau vieler verschiedener Spezialinstrumente, ist kostengünstig und zeitsparend.

Durch die Möglichkeit, mit MUTABOR hochwertige Sampler ansteuern zu können, ist eine Klangqualität zu erreichen, die weit über die Qualität synthetischer Instrumente, die bisher zur Erzeugung von Mikrotönen gebaut wurden, hinausgeht. Die handelsübliche Hardware gestattet momentan eine Intonationsgenauigkeit, die (je nach Instrument) immerhin zwischen 0,8 und 2,4 Cent liegt (Ein Cent entspricht einem Hundertstel eines gleichstufig gestimmten Halbtones).

MUTABOR ist einfach zu bedienen und ohne große Spezialkenntnisse nutzbar. Sie benötigen weder Programmierkenntnisse, noch benötigen Sie Hardwarekenntnisse. Es wird lediglich vorausgesetzt, dass Ihnen die grundlegende Bedienung des Computers bekannt ist. Sie sollten also wissen, wie man den Rechner ein- bzw. ausschaltet. Die „Maus“ sollte bekannt sein, ebenso wie das Einlegen einer Diskette in das Diskettenlaufwerk und das Starten eines Programms. Diese Kenntnisse können Sie dem entsprechenden Bedienungshandbuch des Computers entnehmen.

1.1. Ausführbare Mikrotöne

Eines der großen Probleme bei der praktischen Arbeit mit mikrotonaler Musik ist ihre Ausführbarkeit. Da es sich hierbei um für das „klassische“ Ohr ungewohnte Intervalle handelt, könnte es insbesondere für den praktizierenden Musiker von Nutzen sein, eine genaue Vorgabe des zu spielenden Stückes zu bekommen. Eine mikrotonale Komposition ohne Hörbeispiel und ohne Anleitung vom Komponisten oder einer ähnlich autorisierten Person einfach „vom Blatt“ zu spielen ist extrem schwierig, vielleicht gar unmöglich.

Mit MUTABOR haben Sie ein Instrument erworben, welches es Ihnen ermöglicht, auf beliebigen Tonsystemen bei exakter Intonation zu spielen[1]. Dies könnte als Vorgabe für ein natürliches Instrument dienen, oder als Werkzeug bei der Komposition mikrotonaler Stücke. Insbesondere für vokale Intonationsübungen zur Aufführung von Musikstücken z. B. in barocken Tonsystemen kann MUTABOR dem Interpreten ein wichtiges Hilfsmittel sein.

1.2. Mutierende Stimmungen

Statische Tonsysteme, also Tonsysteme, deren Stimmung sich während eines Musikstückes nicht ändert, lassen sich mit gewissen Einschränkungen auf speziell gestimmten Klavieren realisieren. Ist das Klavier erst einmal gestimmt, so kann man sehr leicht mit dem Tonsystem experimentieren.

Das Konzept vom MUTABOR geht über solche statischen Tonsysteme hinaus. Sie können *mutierende*, also bewegte, „sich verändernde“ Stimmungen programmieren und im freien Spiel anwenden — in Echtzeit. Dies ist bisher auf keinem (mechanischen) Instrument möglich. Die hohe Rechengeschwindigkeit der erweiterten Version 2.1 ermöglicht ein verzögerungsfreies Spielen selbst bei komplexen Stimmungslogiken.

Mutierende Stimmungen ergaben sich bisher fast ausschließlich als Nebeneffekte bei bestimmten Intonationen, z. B. dem „Wandern im tonalen Netz“ bei gewissen Modulationen. Aufgrund ihrer Komplexität und ihrer schwierigen Ausführbarkeit konnten mutierende Stimmungen bisher nicht in der kompositorischen Praxis verwendet werden. MUTABOR II erschließt — bei einfacher Bedienung — nun erstmals einen Teil des Gebietes der mutierenden Stimmungen für den experimentierenden Komponisten.

1.3. Einsatz dieser Dokumentation

Im Lieferumfang von MUTABOR ist dieses Handbuch enthalten. Es gliedert sich in zwei wesentliche Teile, das „Bedienungshandbuch“ und das „Referenzhandbuch“.

Teil I des Bedienungshandbuchs beschreibt die Arbeit mit MUTABOR und gibt eine Anleitung zu Installation und Bedienung der Software. In Teil II erfolgt eine Einführung in die Programmiersprache von MUTABOR, so dass Sie nach dem Durcharbeiten dieses Abschnittes imstande sein sollten, Ihre eigenen Stimmungslogiken zu entwickeln. Wenn Sie mit diesen grundlegenden Programmiertechniken vertraut sind, erfahren Sie weiterführende Konstruktionen im Teil III dieses Handbuches. Das Bedienungshandbuch dient als Lehrgang für den Umgang mit MUTABOR und die Programmierung von Stimmungslogiken. Sie lernen hier die Programmierung von statischen Tonsystemen und von einfachen mutierenden Stimmungen.

Sie werden beim Durcharbeiten des Bedienungshandbuches oft auf Wiederholungen des Stoffes stoßen. Wir halten es aus didaktischen Gründen für sinnvoll, einen

[1] Natürlich im Rahmen der technischen Möglichkeiten, siehe weiter oben. Das aber ist ein Problem des angeschlossenen Synthesizers/Samplers und nicht der Software.

Lehrgang für das Programmieren mit Mutabor nicht so weit in der Information zu reduzieren, dass einmal genannte Fakten als für immer präsent vorausgesetzt werden können. Somit finden Sie an mehreren Stellen im Handbuch z. B. eine Beschreibung, wie man Intervalle deklariert. Falls Sie auf die wiederholte Beschreibungen eines Sachverhaltes treffen, der Ihnen bereits geläufig ist, so lesen Sie einfach darüber hinweg. Der eindeutige Vorteil dieser Methode liegt darin, dass Anwender, die bestimmte Strukturen noch nicht hundertprozentig verstanden haben, diese an verschiedenen Stellen des Handbuches von verschiedenen Standpunkten aus beschrieben bekommen, so dass wir annehmen, dass *wirklich jeder nach dem Lesen des Handbuches in der Lage ist, eigenständig Stimmungslogiken zu programmieren.*

Für die Programmierpraxis und als schnelles Nachschlagewerk dient das „Referenzhandbuch“, in dem alle Befehle der Programmiersprache von Mutabor zusammengefasst und *präzise definiert* sind. Eine Beschreibung der Benutzeroberfläche finden Sie in dem Beiheft „Bedienungsanleitung der Oberfläche“, da das Mutabor – System mittlerweile auf verschiedenen Rechnern implementiert ist, die sich nicht im Kern, sondern nur in der Benutzeroberfläche unterscheiden. Das Benutzerhandbuch ist für alle Mutabor – Implementationen gültig, Abweichungen der einzelnen Versionen, die sich wie gesagt nicht in der Leistungsfähigkeit, sondern nur in der Bedienung unterscheiden, sind in dem Beiheft, welches speziell die Bedienung auf Ihrem Rechner beschreibt, beschrieben. Die Benutzeroberfläche von Mutabor ist sehr einfach und übersichtlich, so dass Sie, falls Sie bereits ein wenig Erfahrung mit anderen Programmen haben, auch ohne das Beiheft zu lesen, sofort mit Mutabor arbeiten können.

1.4. Womit anfangen?

Bevor Sie mit der Installation beginnen, sollten Sie sich das Kapitel „Installation“ in aller Ruhe durchlesen. Für die Installation werden Sie kaum mehr als fünf Minuten benötigen. Wenn die Installation abgeschlossen ist, können Sie im Grunde sofort mit Mutabor arbeiten.

Trotzdem ist es wichtig, dass Sie sich mit der Bedienung von Mutabor vertraut machen. Wenn Sie Ihre eigenen Stimmungslogiken entwickeln wollen, ist es unabdingbar, dass Sie das Kapitel über das „Programmieren von Stimmungslogiken“ sorgfältig studieren.

2. Die Installation der Software

Mutabor ist kostenlos im Internet verfügbar. Sie können das Installationsprogramm auf unserer Homepage unter

`http://www.math.tu-dresden.de/~mutabor/` .

herunterladen.

2.1. Installation auf Festplatte

Sie installieren Mutabor auf Ihrer Festplatte, indem sie einfach das Installationsprogramm ausführen. Mit wenigen Klicks sind sie dann haben sie eine betriebsbereite Version auf Ihrer Festplatte.

2.2. Installation auf anderen Medien

Mutabor ist anspruchslos. Es kann von so ziemlich allen Medien ausgeführt werden, auf denen es Platz findet. Sie können Mutabor mit dem Installationsprogramm auf dem Medium installieren. Möchten Sie ihre Lieblingseinstellungen gleich mit auf dem Medium platzieren, können sie auch gleich den gesamten Ordner kopieren, in dem Mutabor installiert ist. Sie können dann mit dem Medium zu einem anderen Computer wechseln und mit Mutabor arbeiten, ohne es erst installieren zu müssen.

2.3. Anschluss des Synthesizers

Für den Betrieb von Mutabor benötigen Sie noch eine Klangerzeugungseinheit und eine Klaviatur. Sie können zu diesem Zweck jedes MIDI-Keyboard und jeden handelsüblichen Synthesizer oder Sampler benutzen, sofern dafür ein Treiber existiert[1].

Es gibt nun zwei mögliche Konfigurationen: entweder sind Klaviatur und Klangerzeugung in einem Gerät untergebracht, oder Klaviatur (Masterkeyboard) und Klangerzeugung sind getrennt. Wenn Klaviatur und Klangerzeugung in einem Gerät untergebracht sind, so ist es wichtig, dass Sie am Synthesizer/Sampler die Betriebsart „LOCAL OFF“ einstellen, da sonst bei jedem Tastenanschlag zwei Töne erklingen (einmal vom Synthesizer und einmal über MIDI von Mutabor). Schlagen Sie hierzu im Handbuch des Gerätes nach (Stichwort MIDI-Funktionen).

[1] Siehe Abschnitt „Arbeitsweise der Synthesizer- Treiber“

2.3.1. Synthesizer mit Klaviatur

Verbinden Sie nun mit einem MIDI-Kabel die MIDI-OUT-Buchse des Synthesizers mit der MIDI-IN-Buchse am Computer bzw. MIDI-Interface. Dann verbinden Sie die MIDI-IN-Buchse des Synthesizers mit der MIDI- OUT-Buchse am Rechner (bzw. Interface). Damit Mutabor die MIDI-Informationen des Keyboards verstehen kann, ist es wichtig, dass Sie als MIDI- Sendekanal den Kanal 1 an Ihrem Masterkeyboard einstellen.[2]

2.3.2. Separate Klaviatur

Verbinden Sie die MIDI-OUT-Buchse der Klaviatur (Masterkeyboard) mit der MIDI-IN-Buchse des Computers und die MIDI-IN-Buchse des Synthesizers mit der MIDI-OUT-Buchse des Computers. Achten Sie auch hier darauf, dass der Master auf dem MIDI-Kanal 1 sendet!

[2]Genauere Informationen über die MIDI-Spezifikation entnehmen Sie dem Abschnitt „Arbeitsweise der Synthesizer-Treiber“. Die Voreinstellung des MIDI-Kanals 1 kann im Logikprogramm geändert werden. Näheres beschreibt Kapitel 11 »MIDI-Kanäle« im Referenzhandbuch auf Seite 37

3. Mutabor starten

Wenn Sie nun mit MUTABOR arbeiten wollen, so starten Sie einfach das Programm `MUTABOR`, welches sich je nach Konfiguration auf der Festplatte im MUTABOR-Ordner oder auf Ihrer Programmdiskette befindet.[1] Da MUTABOR auf verschiedenen Computern mit unterschiedlicher Bedienungsoberfläche läuft, befindet sich eine genaue Spezifikation der Benutzeroberfläche in dem kleinen Beiheft „Die Benutzeroberfläche“

Um einen einwandfreien Betrieb von MUTABOR zu ermöglichen, ist es sehr wichtig, dass alle Geräte richtig miteinander verbunden sind, und dass der angeschlossene Synthesizer bzw. Sampler für den Betrieb mit MUTABOR eingestellt ist.

Da MUTABOR nur über einen „Trick“ Mikrotöne erzeugen kann, ist es unabdingbar, dass der Synthesizer dafür korrekt eingestellt ist. Leider können wir an dieser Stelle keine allgemeingültige Abfolge von Tasten angeben, die Sie auf Ihrem Gerät drücken müssen, um es für Mikrotöne vorzubereiten. Wir können also nur eine Beschreibung geben, *was* zu tun ist, nicht aber, *wie* es konkret auf Ihrem Gerät einzustellen ist.

Leider haben es die Hersteller handelsüblicher Synthesizer versäumt, in das MIDI-Protokoll Befehle zur direkten Ansteuerung von Mikrotönen einzubauen.[2] Wir müssen daher einen kleinen Trick anwenden, um dem Synthesizer/Sampler trotzdem Mikrotöne entlocken zu können.

Die meisten Synthesizer haben einen „Pitch-Bender“. Das ist ein Drehrad, mit dessen Hilfe man die Frequenz der Töne während des Spiels anheben oder absenken kann. Und zwar je nach Einstellung am Synthesizer zwischen einem und zwölf Halbtönen. Diese Anhebung erfolgt in Schritten von einem Vierundsechzigstel des Maximalintervalls. Wenn man also die Weite des Pitch-Bendings auf einen Halbton einstellt, so kann man damit eine Auflösung von $\frac{1}{64}$ Halbton erreichen, was theoretisch 1,6 Cent entspricht.

Für ein solches Pitch-Bending existiert glücklicherweise auch ein MIDI-Code, mit dem man diese Funktion über MIDI ansteuern kann. Leider verändert eine Pitch-Bend-Nachricht nicht nur einen gezielten Ton, sondern alle momentan liegenden Töne. Auf diese Weise ist es also noch nicht möglich, das c’ um 4 Cent zu erhöhen und gleichzeitig das f” um 35 Cent zu erniedrigen.

[1] Siehe Abschnitt „Installation“

[2] Einzige Ausnahme bildet hier der Synthesizer FB-01 von Yamaha. Hier gibt es eine System-Exklusiv- Meldung der Gestalt „Schalte die Note c mit der Feinstimmung +42 Feinstimmeinheiten an“, wobei eine Feinstimmeinheit einem Hundertachtundzwanzigstel Halbton entspricht. Leider lässt die Klangqualität dieses Gerätes nach heutigen Maßstäben zu wünschen übrig und außerdem wird es nicht mehr produziert. Wahrlich schade, dass Yamaha diese für unsere Zwecke so sinnvolle Option aus ihrem Programm gestrichen hat. Seit Version 3 können wir diesen Treiber leider auch nicht mehr unterstützen.

Doch zu guter Letzt ist auch dieses Problem in den Griff zu bekommen. Wenn man sechzehn Synthesizer hintereinander schaltet, so dass jeder Synthesizer nur *einen* Ton spielt, so könnte man jedem Gerät eine *eigene* Pitch-Bend-Information geben — und das Problem wäre gelöst. Nun hat zwar nicht jeder sechzehn gleiche Synthesizer, aber in diesem Punkt kommt uns die moderne Technik zur Hilfe: die meisten Synthesizer lassen sich in einer „Multi-Mode"-Betriebsart ansteuern. D. h. der Synthesizer tut so, als ob er aus sechzehn eigenständigen Synthesizern gleichen Typs bestehen würde. Man kann nun jeden der sechzehn 'Synthesizer im Synthesizer' gesondert ansprechen, und somit erreichen, dass bis zu sechzehn gleichzeitig liegende Töne jeweils eine *eigene* Feinstimminformation bekommen können.

Dabei sind jedoch folgende Dinge von großer Wichtigkeit:

- Auf jedem MIDI-Kanal muss dasselbe Instrument (Klangfarbe) eingestellt werden, damit mehrere gleichzeitig liegende Töne auch mit derselben Klangfarbe erklingen.

- Auf jedem Kanal muss die Reichweite des Pitch-Benders auf 1 Halbton eingestellt werden, damit die Feinstimmungen korrekt durchgeführt werden, denn eine Feinstimmungseinheit entspricht $\frac{1}{64}$ Weite. Diese „Pitch Bend Range" ist bei den meisten Synthesizern ein Parameter der Klangfarbe und nicht des Multis.

Diese Einstellungen erfordern gewisse Kenntnisse über die Bedienung des benutzten Synthesizers, über die dieses Handbuch aufgrund der Mannigfaltigkeit des Angebotes von Synthesizern und Samplern leider keine genaue Beschreibung liefern kann. Glücklicherweise sind nur die beiden oben genannten Einstellungen nötig, und wenn Sie erst einmal wissen, wie Sie Ihren Synthesizer richtig einzustellen haben, dürfte dies kein Problem mehr darstellen. Falls Sie dennoch mit der Einstellung Ihres Synthesizers Schwierigkeiten haben, so stehen wir Ihnen gerne mit Rat und Tat beiseite, denn an solchen technischen Kleinigkeiten sollte der Einsatz von MUTABOR nicht scheitern.

Nachdem Sie die Software installiert haben und alle Geräte richtig miteinander verbunden und eingestellt sind, können Sie mit MUTABOR arbeiten.

Außerdem befindet sich auf der Originaldiskette ein Ordner, in dem sich die Quelltexte diverser Stimmungslogiken befinden, unter anderem auch viele in diesem Handbuch aufgeführte Beispiele. Diese Beispiele können direkt benutzt werden, oder als Vorlage dienen, um zu experimentieren. Man kann diese Beispiele mit dem Text-Editor verändern und sich danach die geänderten Beispiele anhören.

4. Die Entwicklung von Logikprogrammen

4.1. Stimmungslogiken programmieren

Irgendwann wollen Sie sicherlich eigene Stimmungslogiken entwickeln oder vielleicht die Demo-Stimmungen modifizieren. Jede Stimmungslogik wurde in einer speziell entworfenen Programmiersprache für Stimmungslogiken programmiert. Man formuliert in der Programmiersprache eine Folge von Anweisungen, die den Computer im Laufzeitmodul veranlassen, den Synthesizer richtig zu stimmen. Man schreibt also ein *Programm*. Wenn der Programmtext einmal auf dem Papier entworfen worden ist, muss er in den Computer eingegeben und auf einem Datenträger wie Diskette oder Festplatte gespeichert werden. Dies geschieht mit Hilfe eines *Text-Editors*. Genauso wie Sie mit einer Textverarbeitung Briefe schreiben können, werden Sie mit dem Texteditor den Programmtext für Ihre Logiken eingeben und verändern. MUTABOR verfügt über einen programminternen Texteditor, der den Bedürfnissen des Programmierens genügt.

Wenn dies geschehen ist und Sie Ihre Logik gerne im freien Spiel anwenden möchten, ist es notwendig, dass der Programmtext vom sogenannten „Compiler“ in interne Tabellen übersetzt wird. Das Logikprogramm, welches Sie mit dem Texteditor eingegeben und abgespeichert haben, besteht nur aus einer Ansammlung von für den Computer bedeutungslosen Zeichen. Um diesen Text „verstehen“ zu können, und insbesondere um Ihre Stimmungslogik ausführen zu können, muss dieser „Quelltext“ übersetzt werden. Diese Aufgabe übernimmt der *Compiler*. Während des Übersetzens überprüft der Compiler automatisch Ihr Programm auf seine syntaktische Richtigkeit. Falls der Quelltext fehlerhaft ist, wird der Übersetzungsvorgang abgebrochen und auf dem Bildschirm erscheint eine Fehlermeldung, die Ihnen anzeigt, wo der Fehler gefunden wurde, und welcher Natur er ist. Sie sollten daraufhin wieder den Editor aufrufen, den Fehler verbessern und einen erneuten Übersetzungsvorgang starten.

Wenn der Compiler keinen Fehler meldet, können Sie nun direkt mit der Stimmungslogik experimentieren. Und hier noch einmal die drei Schritte der Programmentwicklung auf einen Blick:

- Entwerfen Sie auf dem Papier den Programmtext Ihrer Stimmungslogik.

- Geben Sie ihn im Texteditor ein und speichern Sie die Datei auf Diskette oder Festplatte.

- Compilieren Sie das Logikprogramm in MUTABOR und aktivieren Sie es.[1]

4.2. Die Schritte einer Programmentwicklung

An einem Beispiel sollen Sie nun lernen, wie Sie ein Logikprogramm entwickeln und zum Laufen bringen können. Nehmen wir an, Sie hätten folgendes Logikprogramm geschrieben und wollten mit dieser Logik experimentieren[2]:

```
INTERVALL
    Drittelton = 18 Wurzel 2
TON
    a = 440
TONSYSTEM
    Drittel = 69 [a] Drittelton
LOGIK
    Drittel Taste D = Drittel [ ]
```

(Dies ist ein einfaches Beispielprogramm; den Umgang mit der Programmiersprache von MUTABOR lernen Sie in Teil II und III dieses Handbuches.)

4.2.1. Datei auswählen

Jedes Logikprogramm, das Sie auf MUTABOR programmieren, wird unter einem beliebigen Dateinamen (z. B. mit der Erweiterung `.mut`) auf Diskette oder Festplatte gespeichert[3]. Im Laufe Ihrer Arbeit mit MUTABOR werden Sie also verschiedene Logikprogramme geschrieben haben, die sich dann auf einem Speichermedium (Diskette oder Festplatte) befinden.

Bevor Sie nun mit Ihrem Logikprogramm im freien Spiel experimentieren können, muss der Quelltext auf seine grammatikalische Richtigkeit überprüft und in die computerinternen Stimmungstabellen umgewandelt werden. Diese Aufgabe übernimmt der Compiler. Sie müssen lediglich den Menüpunkt „Logik laden“ anklicken und einen kleinen Moment warten[4].

Wenn Ihr Programm fehlerfrei übersetzt werden konnte, so meldet dies MUTABOR und man kann die Logik starten. Wenn Ihr Quelltext an einer Stelle syntaktisch

[1] Der Ladevorgang selbst wird mit Hilfe des Compilers durchgeführt.

[2] Viele im Handbuch abgedruckte Logikprogramme finden Sie auch auf der Installationsdiskette oder im Internet. Sie brauchen also die wichtigsten Beispiellogiken aus dem Handbuch nicht mühsam ‘abzutippen’. In einer Fußnote wird jeweils der Dateiname angegeben; dieses Beispiel finden Sie unter dem Dateinamen `drittel.mut`

[3] Ihr Computer verwaltet Dateien unter einem Dateinamen, welcher aus zwei Teilen besteht. Beide Teile sind durch einen Punkt voneinander getrennt, z. B. `lernen.doc`. Den ersten Teil nennen wir „Namen“, den zweiten Teil „Erweiterung“. Der Name darf bei einigen Rechnern maximal acht Zeichen enthalten, muss mit einem Buchstaben beginnen und darf danach auch Ziffern oder das „Tiefstrich“-Zeichen (_) enthalten. Die Erweiterung darf bei einigen Systemen maximal drei Zeichen umfassen.

[4] Bei größeren Programmen oder einer langsamen Festplatte oder Diskettenbetrieb kann dieser „kleine Moment“ manchmal zu einem „großen Moment“ werden …

falsch ist, macht Sie der Compiler auf den Fehler aufmerksam und unterbricht seine Arbeit. Sie können nun erneut den Texteditor aufrufen, den Fehler verbessern und den Compiler zu einem neuen Übersetzungsversuch anklicken. Die häufigsten Programmierfehler und deren Beseitigung finden Sie im Anhang „Fehlerursachen“. Nebenbei sei bemerkt, dass der Compiler nur den ersten gefundenen Fehler meldet und dann die Übersetzung abbricht. Man muss diesen Fehler korrigieren und nochmal übersetzen, bis keine Fehler mehr gemeldet werden. Erst wenn ein Logikprogramm ohne Fehlermeldung übersetzt werden konnte, kann man es ausprobieren.

Nachdem Ihr Logikprogramm erfolgreich übersetzt und somit geladen wurde, steht der Ausführung nichts mehr im Wege. Auf dem Bildschirm erscheint das Laufzeitmodul und Sie können Ihre Stimmungslogiken im freien Spiel anwenden.

Teil II.

Beschreibung der Programmiersprache von Mutabor

Dieser Teil des Handbuches gibt Ihnen eine Einführung in das Programmieren von Stimmungslogiken. An vielen Beispielen werden Sie die Programmierfunktionen von Mutabor kennen lernen. Solche Beispiele sind immer in einer Schreibmaschinenschrift gedruckt, z. B. :

```
INTERVALL
    Quinte = 3:2
```

Diese Schrift bedeutet, dass es sich um Programme oder Programmfragmente handelt, die Sie genau so, wie sie im Handbuch stehen, im Editor eingeben können.

Die Programmiersprache von Mutabor hat drei verschiedene Grundelemente: *Tonsysteme, Umstimmungen* und *Logiken*. Diese drei Elemente seien im folgenden beschrieben. Beachten Sie, dass wir für eine Programmiersprache präzise definieren müssen, was wir unter einem Tonsystem (Umstimmung, Logik, . . .) verstehen wollen. In der Musiktheorie wird der Begriff des Tonsystems keineswegs einheitlich verstanden. Hier jedoch ist es nötig, den Begriff des Tonsystems mathematisch exakt zu definieren. Diese Definition ist so geartet, dass sie dem intuitiven Verständnis möglichst nahe kommt. Es soll also mit dieser Definition für die MUTABOR-Programmierzwecke eine Art „Mutabor-Tonsystem“ geschaffen werden, welches den hier angegebenen Regeln genügt und als eindeutig definierter Begriff benutzbar ist. Des intuitiven Verständnisses wegen behalten wir im folgenden den Begriff „Tonsystem“ bei und meinen damit immer das hier definierte „MutaborTonsystem“. Diese Präzisierung gilt sinngemäß auch für alle anderen Elemente der Programmiersprache (Umstimmung, Logik, . . .). **Wir wollen diese Begriffe keineswegs generell für die Musiktheorie so definiert wissen, sondern lediglich innerhalb dieser formalen Logiksprache eine Basis schaffen, mit der eine eindeutige Stimmungslogik programmierbar wird.**

Kommentare können an beliebigen Stellen in einem Logikprogramm geschrieben werden und dienen dem menschlichen Leser zur Dokumentation des Programms. Kommentare stehen in doppelten Anführungszeichen

```
" Dies ist ein Kommentar "
```

können sich über mehrere Zeilen erstrecken und haben keinerlei Einfluss auf die sonstige Funktionsweise des Logikprogramms.

5. Tonsysteme

Ein Tonsystem ist eine statische, d. h. unveränderliche Stimmung. Mit einem bestimmten System werden alle Tasten der Klaviatur wie bei einer Klavierstimmung gestimmt, d. h. jeder Taste der Klaviatur wird eine bestimmte Frequenz zugewiesen, einschließlich der Möglichkeit, dass eine Taste keinen Ton hervorbringt.

5.1. Die vier Parameter eines Tonsystems

Jedes Tonsystem, das mit Mutabor programmiert werden kann, lässt sich durch genau vier Parameter exakt beschreiben.

5.1.1. Die Töne der Fundamentaltonleiter

Mit Fundamentaltonleiter (FT) bezeichnen wir einen bestimmten zusammenhängenden Ausschnittsbereich der Klaviatur. Dieser Bereich kann beliebig viele Tasten umfassen und an jeder beliebigen Stelle der Klaviatur beginnen; zum Beispiel die Tasten vom eingestrichenen c' bis zum eingestrichenen h', oder nur eine Taste, usw.. . . Jeder Taste dieses Bereiches wird eine Frequenz zugewiesen, die beim Anschlagen derselben erklingen soll (einschließlich der Möglichkeit, dass eine Taste keinen Ton erzeugt, also stumm geschaltet ist).

5.1.2. Die Breite der FT

Die Anzahl an Tasten, welche die Fundamentaltonleiter darstellen, nennen wir Breite der FT.

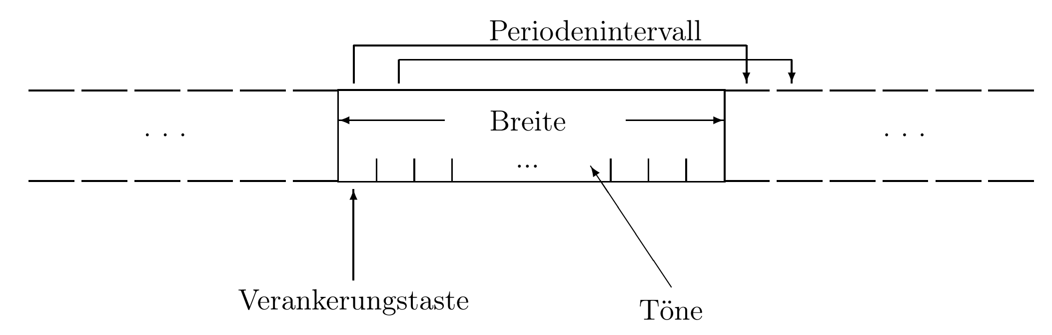
5.1.3. Die Verankerungstaste

Die Taste der Klaviatur, auf der die FT beginnt, nennen wir Verankerungstaste (Es ist natürlich auch möglich, dass die Verankerungstaste keinen Ton erzeugt). Der MIDI-Standard hat alle Tasten der Klaviatur nummeriert. Das eingestrichene c' hat die Nummer 60, das eingestrichene a' die Nummer 69 (a' liegt 9 Tasten bzw. Halbtöne über c').

5.1.4. Das Periodenintervall

Bisher haben wir nur einen kleinen Ausschnitt der Klaviatur mit Tönen, d. h. Frequenzen belegt. Wir benötigen also noch eine Art Reproduktionsvorschrift, wie man

aus der FT die komplette Frequenzbelegung der Klaviatur berechnen kann. Dies geschieht mit einem einzigen Parameter, dem „Periodenintervall“. Gewöhnlicherweise soll die Struktur der Fundamentaltonleiter *blockweise wiederholt* werden, und zwar um ein bestimmtes Intervall verschoben. Dieses Intervall nennen wir das „Periodenintervall“. Somit haben wir nun alle vier Parameter besprochen, die ein Tonsystem definieren. Diese sehr einfache Definition des Begriffes „Tonsystem“, die zunächst aus einer Verallgemeinerung des Begriffes der Tonleiter entstanden ist, bietet dem Anwender bereits eine unglaubliche Vielfalt an Möglichkeiten, das Instrument zu stimmen. Selbstverständlich gibt es darüber hinaus noch statische mikrotonale Strukturen, die sich nicht in dem Schema Verankerungstaste – Töne – Periodenintervall einordnen lassen, solche Systeme lassen sich aber behelfsweise durch ein Tonsystem mit der Breite der gesamten Klaviatur programmieren, so dass wirklich jede Taste eine individuelle Frequenz zugewiesen bekommt. Glücklicherweise basieren die meisten herkömmlichen Tonsystemstrukturen auf unserem Prinzip der „Fundamentaltonleiter“ und sind somit sehr leicht zu realisieren. Die folgende Grafik soll noch einmal den Aufbau eines Tonsystems aus den vier Parametern *Verankerungstaste, Töne, Breite* und *Periodenintervall* veranschaulichen:



5.2. Eine mathematische Betrachtung

Die Definition dieses Tonsystems lässt sich mathematisch exakt formulieren:

Es seien α die Verankerungstaste des Tonsystems (als MIDI- Nummer), δ die Breite und ψ das Periodenintervall. Außerdem sei eine Funktion $\xi(\tau)$ mit $0 \leq \tau \leq (\delta - 1)$ definiert, die als Ergebnis die Frequenz des τ-ten Tones der Fundamentaltonleiter liefert. Durch diese vier Parameter wird das Tonsystem vollständig beschrieben. Wenn nun die (MIDI-)Taste t gedrückt wird, so muss der Synthesizer eine Frequenz von y Hertz spielen, wobei sich y als Funktion von α,δ,ψ,ξ und t wie folgt schreiben lässt:

$$y = \xi((t - \alpha) \bmod \delta) \cdot \psi^{\,(t-\alpha)\mathrm{div}\ \delta}$$

$(t - \alpha)\mathrm{div}\delta$ entspricht dem Abstand der Taste von der FT[1], gemessen in Einheiten der FT-Breite und $(t - \alpha) \bmod \delta$ steht für die Taste der FT, die der angeschlagenen

[1] *mod* bedeutet hier den Rest bei der Teilung, z. B. 24 *mod* 7 = 3; *div* bedeutet ganzzahliger Quotient der Teilung, z. B. 14 *div* 5 = 2

entspricht, also eine Art Tonigkeit. Anschaulicher lässt sich die Formel schreiben als:

$$y = \xi(\text{Bezugston}) \cdot \psi^{\text{Abstand zur FT}}$$

Die intervallischen Zusammenhänge zwischen den Tönen der Fundamentaltonleiter nennen wir *Intervallstruktur*.

5.3. Hilfsmittel: Intervalle und Töne

Bisher musste die Erstellung der Grundparameter eines Tonsystems mühsam mit dem Taschenrechner geschehen. Die Frequenzen der Fundamentaltöne mussten berechnet werden; alle Werte als Kommazahlen eingegeben werden, ... um Ihnen diese mühsame Arbeit zu ersparen, und um die Struktur eines programmierten Tonsystems deutlich erkennbar zu machen, stellt Ihnen die Programmiersprache von MUTABOR zwei wichtige Hilfsmittel zur Verfügung:

5.3.1. Intervalldeklarationen

In einem Tonsystem stehen die Töne der Fundamentaltonleiter (FT) in einer bestimmten Intervallbeziehung zueinander. Die Aussage „G liegt um eine Quinte höher als C“ bedeutet, dass man die Frequenz von „C“ mit dem Faktor „Quinte“ multiplizieren muss, um die Frequenz von „G“ zu erhalten. Da solche Intervalle für die Definition von Tonsystemen von so grundlegender Bedeutung sind, können Sie sich in einem MUTABOR-Programm Ihre Intervalle selbst definieren. Ein paar Beispiele:

```
INTERVALL
    Quinte     = 3 : 2
    Terz       = 5 : 4
    Halbton    = 12 Wurzel 2
    Oktave     = 2 : 1
    Drittelton = 18 Wurzel 2
    Cent       = 1200 Wurzel 2
    Syn_Komma  = 4 Quint - 2 Oktave - Terz
```

Zulässige Zuweisungen sind: Proportionen (z. B. `32:19`, aber auch `3.14159:4`) und Wurzelausdrücke (z. B. `9 Wurzel 1.34`[2]). Komplexe Ausdrücke, also Intervalldeklarationen, die sich auf andere Intervalle beziehen (siehe letzte Deklaration im Beispiel), sind ebenfalls erlaubt, genaueres darüber lesen Sie im Anhang A.

5.3.2. Tondeklarationen

Nachdem Sie nun alle Intervalle, auf die Ihr Tonsystem aufbaut, mit einem Namen und einem Wert versehen haben, können Sie die Zusammenhänge zwischen den Tönen

[2] 9 Wurzel 1.34 ist zu lesen als 9te Wurzel aus 1.34 ($\sqrt[9]{1.34}$)und nicht 1.34te Wurzel aus 9

bestimmen. Nehmen wir das Beispiel von oben: „G liegt um eine Quinte höher als C“. Das würde man ganz einfach so formulieren:

```
TON
    G = C + Quinte
```

Bei den Tondeklarationen werden also die *intervallischen Zusammenhänge* zwischen den Tönen definiert. Die Vorgehensweise ist analog der des Klavierstimmens. Man beginnt das Klavierstimmen normalerweise mit der Festlegung des Kammertones. Somit haben Sie auch bei MUTABOR die Möglichkeit, einem[3] Ton einen festen Frequenzwert zuzuweisen, z. B.

```
TON
    a = 440
```

Hier wird dem Ton `a` eine Frequenz von `440` Hertz zugewiesen. Solche „absoluten“ Zuweisungen sind insofern von elementarer Bedeutung, als innerhalb der Tondeklarationen immer *mindestens ein Ton* eine Absolutfrequenz zugewiesen bekommen muss, da das Tonsystem sonst nicht eindeutig bestimmt ist. Wenn Sie z. B. nur die Töne C und G wie folgt bestimmen:

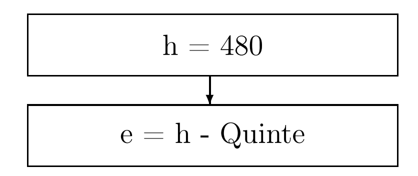
```
TON
    C = G + Quinte
    G = C - Quinte
```

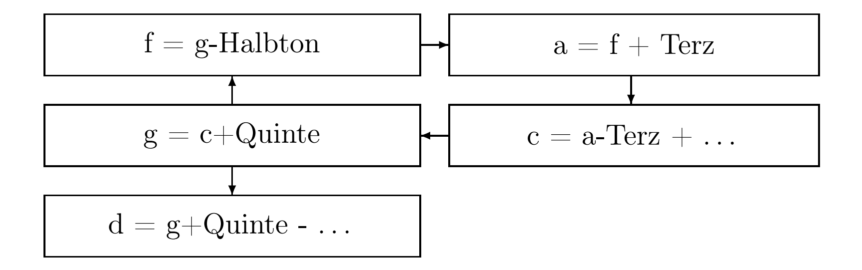
und keine weiteren Töne deklarieren, so ist ein Tonsystem, welches diese Töne benutzt, nicht eindeutig bestimmt (C und G hängen ohne Fußpunkt endlos-rekursiv voneinander ab). Die Gesamtheit aller Tondeklarationen darf keine solche „Zyklen“ enthalten. Somit wäre auch folgende Sammlung von Tönen nicht konsistent und würde eine Fehlermeldung des Compilers hervorrufen:

```
TON
    a = f + Terz
    c = a - Terz + Quinte - Oktave
    g = c + Quinte
    f = g - Halbton
    d = g + Quinte - Oktave
    h = 480
    e = h - Quinte
```

Zwar ist hier ein Ton absolut deklariert (`h=480`), aber die Töne a,c,g,f und d sind geschlossen miteinander verkettet und somit nicht eindeutig bestimmt, was Sie dem folgenden Diagramm entnehmen können:

[3] oder auch mehreren verschiedenen Tönen





In diesen Deklarationen steckt noch ein zweiter gravierender Fehler. Die Töne c,g,a und f hängen zyklisch voneinander ab. Wenn Sie Ihre Tondeklarationen in Form solcher Graphen darstellen, erkennen Sie sofort eventuelle Fehler. Nicht eindeutig bestimmte Töne zeichnen sich dadurch aus, dass sie keine Verbindung zu einem Ton mit Absolutfrequenz haben. Dies ist, als ob Sie Ihrem Klavierstimmer zwar sagen, wie er von einem Ton zum anderen stimmen soll, ihm aber nicht sagen, wo er anfangen soll ... Es ist wichtig zu bemerken, dass die Reihenfolge, in der die Töne deklariert werden völlig bedeutungslos ist. Im Gegensatz zu vielen Programmiersprachen, wie Pascal, ... gilt bei Mutabor nicht das Prinzip „erst deklarieren, dann benutzen". Wichtig ist nur, dass die Verknüpfung der Töne in sich konsistent ist. Der große Vorteil dieser Tondeklarationen besteht einfach darin, dass der *strukturelle Zusammenhang* der Töne unmittelbar – und auch für jemanden, der noch nie mit Mutabor II gearbeitet hat – ersichtlich wird.

Desweiteren können solche Tonbezüge auch faktorielle Zusammensetzungen von Intervallen enthalten, z. B.

```
TON
    c' = 264
    e' = c' + 4 Quinte - 2 Oktave
```

Die Frequenz des Tones e' berechnet sich also[4] aus der Frequenz der Tones c' (= 264 Hz) , von dort aus vier Quinten nach oben (= 1336,5 Hz), und dann wieder zwei Oktaven nach unten. e' erklingt also hier mit $334\frac{1}{8}$ Hz.

5.3.3. Zusammenfassung

Wie in der Einleitung bereits gesagt wurde, ist es beim Entwurf einer Programmiersprache – wie überhaupt im Umgang mit Computern – absolut notwendig, eine

[4] Wir nehmen an, dass die Intervalle `Quinte` und `Oktave` wie in den vorigen Beispielen als `3:2` bzw. `2:1` deklariert wurden.

präzise Sprache zu sprechen, also nur mit *wohldefinierten Begriffen* zu hantieren. Der Computer kann nur Stimmungslogiken „verstehen“, deren Programmtext sich 100%ig an die vorgegebene Grammatik (Syntax) der Sprache hält. Damit dem Leser die Syntax der Programmiersprache von Anfang an verinnerlichen kann, sind die Beispiele einerseits zwar an der Basis musiktheoretischer Begriffe orientiert, sie sollen andererseits aber auch einen Abstraktheitsgrad erreichen, dass mittels einer abstrakten Syntax auch eine völlig neue musikalische Semantik geschaffen werden kann.

Doch zurück zu konkreten Tönen. Die verschiedenen Möglichkeiten einer Tondeklaration lauten:

a) Absolutfrequenz in Hertz, z. B. `XX = 546.3324`

b) Verknüpfung, genauer: Bezugston ± Distanz, z. B.
`XX = YY + 3 Quinte - 4 Terz + 17 Oktave` oder
`ZZ = QQ - Halbton`

5.4. Tonsystem-Deklaration

Nachdem mit der Deklaration von Tönen das Fundament des Tonsystems gelegt wurde, kommen wir nun zu dessen Deklaration: Jedes Tonsystem bekommt – genauso wie Intervalle und Töne – einen Namen. Auf diesen folgen die Parameter Verankerungstaste, Töne der FT und zuletzt das Periodenintervall (Die Breite der FT ist implizit durch die Anzahl der Tasten in der FT gegeben.). Ein Beispiel:

```
TONSYSTEM
    C_Dur = 60 [ c,des,d,es,e,f,fis,g,as,a,b,h ] Oktave
```

Dieses Tonsystem ist auf der Taste 60 (=eingestrichenes c’) verankert, die FT enthält die Töne c,des,d, . . . , h und das Periodenintervall ist die Oktave. Alle Töne und das Periodenintervall müssen irgendwo im Programm definiert werden; die Benutzung eines nicht definierten Bezeichners führt zu einer Fehlermeldung des Compilers. Wenn Sie in Ihrem Programm Intervalle, Töne und ein Tonsystem deklariert haben, es mit dem Compiler übersetzt haben und aufrufen, so ist Ihr Tonsystem trotzdem noch nicht aufrufbar. Wenn Sie es hören bzw. spielen wollen, so müssen Sie dem Programm noch die Zeilen

```
LOGIK
    Name Taste x = Name [ ]
```

hinzufügen, wobei Sie statt „Name“ den tatsächlichen Namen Ihres Tonsystems einsetzen und statt ’x’ irgendeinen Buchstaben. Die Oberfläche auf dem Bildschirm zeigt dann in einer „Tastenliste“ an, dass die Taste x die Logik *Name* aktiviert. So können Sie im Laufzeitmodul Ihre Stimmungslogiken (bzw. indirekt Ihre Tonsysteme) per Tastendruck aktivieren. (Den Sinn dessen lesen Sie im Abschnitt „Logiken“ bzw. „Anweisungen, Auslöser und Aktionen“) .

5.5. Beispiele für Tonsysteme

5.5.1. 'Reines' C-Dur

Dieses Programm beschreibt einen Ausschnitt aus dem tonalen Netz (siehe Abschnitt „Beschreibung der Demonstrationslogiken“) mit der Tonigkeit C als Zentrum[5].

```
INTERVALL
    Quinte = 3 : 2
    Terz   = 5 : 4
    Oktave = 2 : 1
TON
    c   = a - Terz + Quinte - Oktave
    des = f - Terz
    d   = g + Quinte - Oktave
    es  = g - Terz
    e   = c + Terz
    f   = c - Quinte + Oktave
    fis = d + Terz
    g   = c + Quinte
    as  = c - Terz + Oktave
    a   = 440
    b   = c - 2 Quint + 2 Oktave
    h   = g + Terz
TONSYSTEM
    C_Dur = 60 [ c,des,d,es,e,f,fis,g,as,a,b,h ] Oktave
LOGIK
    C_Dur Taste C = C_Dur [ ]
```

5.5.2. Einige gleichschwebende Temperaturen

Trotz der weit verbreiteten Sprechweise von „gleichschwebenden“ Temperaturen ist es korrekter, von *gleichstufigen* Stimmungen zu sprechen. Gleichstufige Teilungen der Oktave in beliebig viele Stufen sind besonders einfach zu programmieren[6]. In manchen Fällen macht einem aber die 12er – Periodizität der Klaviatur zu schaffen. Dieses Problem wird im nächsten Beispiel angegangen.

```
LOGIK
    Halb Taste H = Halb [ ]
    Drittel Taste D = Drittel [ ]
    Viertel Taste V = Viertel [ ]
TONSYSTEM
    Halb = 69 [a] Halbton
```

[5]Beispielprogramm c_dur.mut
[6]Beispielprogramm gleich.mut

```
    Drittel = 69 [a] Drittelton
    Viertel = 69 [a] Viertelton
TON
    a=440
INTERVALL
    Halbton = 12 Wurzel 2
    Viertelton = 24 Wurzel 2
    Drittelton = 18 Wurzel 2
```

An diesem Beispiel sehen Sie, dass die Reihenfolge, in der Sie die Deklarationsteile schreiben, keine Rolle spielt.

5.5.3. Eine pentatonische Skala

Hier sehen Sie zwei verschiedene Ausführungen des gleichen Tonsystems, welche sich nur durch unterschiedlichen Spielkomfort unterscheiden. Die Fundamentaltonleiter dieses (gleichschwebend) pentatonischen Tonsystems umfasst fünf Töne. Die erste Logik „PENTA1“ realisiert dies zwar sehr einfach, aber die Spielbarkeit leidet spürbar unter dieser Einfachheit. Das Problem ist hier die auf zwölfstufige Tonsysteme ausgelegte Klaviatur. Die Oktave liegt zwischen a', d', g' - und bevor man mit dieser Tastenbelegung umgehen kann, sind doch einige Umgewöhnungen nötig.

Dieses Problem ist in der zweiten Logik „PENTA2“ beseitigt. Hier sind nur die schwarzen Tasten der Klaviatur belegt - die pentatonische Skala ist einwandfrei spielbar (Die 440 Hertz liegen jetzt auf der Taste cis'=61). Wenn Sie in einem Tonsystem bestimmte Tasten nicht belegen wollen, also stumm schalten möchten, so lassen Sie einfach den Platz zwischen den Kommata frei[7].

```
LOGIK
    PENTA1 Taste P = Penta1 [ ]
    PENTA2 Taste Q = Penta2 [ ]
TONSYSTEM
    Penta1 = 69 [a] Pentaton
    Penta2 = 60 [,a,,next1,,,next2,,next3,,next4,] Oktave
INTERVALL
    Pentaton = 5 Wurzel 2
    Oktave   = 2 : 1
TON
    a = 440
    next1 = a + Pentaton
    next2 = a + 2 Pentaton
    next3 = a + 3 Pentaton
    next4 = a + 4 Pentaton
```

[7] Beispielprogramm penta.mut

Betrachten wir nochmals das Tonsystem „Penta2“. Auf der Verankerungstaste (60=c') liegt kein Ton. Die darauf folgende Taste (61=cis') bekommt den Ton a, etc. Wichtig ist der leere Platz vor der geschlossenen eckigen Klammer. Hätten wir einfach

```
Penta2 = 60 [,a,,next1,,,next2,,next3,,next4] Oktave
```

geschrieben, so würde sich die FT bereits auf der Taste h' um eine Oktave versetzt wiederholen und wir hätten noch mehr Chaos als bei der Logik „PENTA1“. Wenn Sie also eine spezielle Tastenbelegung für Ihr Tonsystem wünschen, so ist es wichtig, genau zu berücksichtigen, welche Tasten zur Fundamentaltonleiter gehören, und welche nicht.

6. Umstimmungen

6.1. Grundlage für mutierende Stimmungen

Kommen wir nun zum eigentlich interessanten Teil von MUTABOR, der Programmierung von *mutierenden Stimmungen*. Hierbei handelt es sich, wie in der Einleitung bereits beschrieben, um Stimmungen, bei denen die Taste-Frequenz-Zuordnung nicht fest ist, sondern im Laufe des Spiels verändert wird.

MUTABOR kann solche Veränderungen der Stimmung, kurz: *Umstimmungen*, nicht nur von außen steuerbar machen (z. B. Knopfdruck, MIDI-Ereignis, ...), sondern Umstimmungen als Reaktion auf die gespielten Harmonien definieren. Doch bevor wir auf solche Umstimmautomatiken eingehen, soll zunächst der Umstimmvorgang selbst genau betrachtet werden.

Sie können in einer Umstimmung jeden der vier Tonsystem-Parameter verändern. Veränderungen von Verankerungstaste und/oder Breite der Fundamentaltonleiter verändern zusätzlich andere Parameter, um gewisse Strukturmerkmale des Tonsystems sinnvoll zu transformieren. Diese beiden Umstimmungsarten sind deshalb trotz ihres einfachen Aufrufs eigentlich komplexere Umstimmungen.

6.1.1. Das aktuelle Tonsystem

Die momentane Stimmung von MUTABOR lässt sich auch bei mutierenden Stimmungen als Tonsystem charakterisieren. Diesen Zustand nennen wir „aktuelles Tonsystem“. **Das „aktuelle Tonsystem“ ist also die Stimmung von MUTABOR im aktuellen Moment.** Diese Stimmung kann explizit im Programm angegeben sein (als Tonsystemdeklaration). In den meisten Fällen aber wird sich das „aktuelle Tonsystem“ als Produkt einer Folge von relativen Umstimmungen ergeben und nirgendwo explizit angegeben worden sein.

Wenn im Zusammenhang von Umstimmungen und später Stimmungslogiken vom „alten Tonsystem“ die Rede ist, so ist das Tonsystem gemeint, das unmittelbar vor der Umstimmung das „aktuelle Tonsystem“ war.

6.1.2. Die zwei Umstimmtypen

Da der Begriff des Tonsystems bereits eingeführt ist, können wir eine Umstimmung wie folgt definieren: *Eine Umstimmung ist der Wechsel von einem Tonsystem in ein anderes.* Das aktuelle Tonsystem wird verändert, es „mutiert“. Dieser Wechsel kann auf zweierlei Arten vonstatten gehen, und dies unterscheidet die zwei grundlegenden Umstimmtypen bei MUTABOR:

- Absolute Umstimmungen. Hier wird der alte Zustand des Tonsystems nicht berücksichtigt, und ohne jeden Zusammenhang einfach alle (bzw. ein) Parameter neu gesetzt. Eine absolute Umstimmung wäre es z. B., wenn der dritte Ton der FT auf 375,8 Hz gestimmt, oder das Periodenintervall auf „Septime" gesetzt wird. Dabei spielt es keine Rolle, welchen Wert der Parameter vor der Umstimmung hatte. (Daher der Begriff „absolute" Umstimmung)

- Relative Umstimmungen. Hier wird der neue Wert des Parameters aus dem aktuellen Zustand berechnet. Die Zuweisung ist also nicht absolut oder kontextfrei. Der neue Zustand lässt sich als „relativ" zum alten beschreiben. Eine relative Umstimmung wäre es z. B., wenn die Verankerungstaste um eine Taste angehoben wird, oder der fünfte Ton der FT um ein syntonisches Komma erniedrigt wird. Relative Umstimmungen ermöglichen es Ihnen, „auf endlich vielen Tasten unendlich viele Töne spielen zu können"

6.2. Absolute Umstimmungen

Die Syntax der Umstimmungen ist an die Deklaration des Tonsystems angelehnt. Dort steht *vor* den eckigen Klammern die Verankerungstaste, *in* den Klammern die Töne der FT und *hinter* den Klammern das Periodenintervall. Wenn Sie in einer Umstimmung die Verankerungstaste verändern wollen, so schreiben Sie den Umstimmausdruck *vor* eckige Klammern; Veränderungen der Töne gehören *zwischen* eckige Klammern, ebenso eine Veränderung der Breite der FT; soll das Periodenintervall geändert werden, so steht der Ausdruck *hinter* eckigen Klammern. (Dies gilt für absolute und relative Umstimmungen gleichermaßen.)

Sie können in einer Umstimmung jeden der vier Grundparameter verändern. Welche Konsequenzen solche Umstimmungen jeweils haben und wie sie genau formuliert werden können, lesen Sie in den folgenden Abschnitten.

6.2.1. Verändern des Periodenintervalls

Eine Änderung des Periodenintervalls hat keine Auswirkungen auf die Töne der FT, es werden also nur diejenigen gedrückten Tasten umgestimmt, die nicht im Bereich der FT liegen. Probieren Sie folgende Logiken einmal aus[1] und schalten Sie während des Spiels zwischen ihnen hin und her[2]. Sie werden deutlich hören, wie sich das Periodenintervall ändert. (Da Logiken noch nicht besprochen wurden, hier zur Erklärung: Eine Logik der Gestalt

```
LOGIK
    Name Taste x = Umstimmungsname [ ]
```

[1] D. h. Eingabe mit dem Editor, Speichern (Text-Datei), vom Compiler übersetzen lassen und aufrufen.

[2] Indem Sie die Tasten E,Z und D drücken.

führt bei ihrem Aufruf durch Drücken der Taste x (einmalig) die angegebene Umstimmung durch.) Doch nun das Programm[3]:

```
INTERVALL
    Erstens = 12 Wurzel 2
    Zweitens = 13 Wurzel 2
    Drittens = 14 Wurzel 2
UMSTIMMUNG
    E = [ ] Erstens
    Z = [ ] Zweitens
    D = [ ] Drittens
LOGIK
    Eins Taste E = E [ ]
    Zwei Taste Z = Z [ ]
    Drei Taste D = D [ ]
```

Sie werden sich bei diesem Beispiel wahrscheinlich fragen, mit welchem Tonsystem das Laufzeitmodul initialisiert wird, da hier kein Tonsystem angegeben ist. Hierbei gilt grundsätzlich: *Das Laufzeitmodul wird immer mit einer gleichstufigen Halbtontemperierung initialisiert*, also mit

```
INTERVALL Halb = 12 Wurzel 2
TON a = 440
TONSYSTEM Start = 69 [ a ] Halb
LOGIK keine_Logik = Start [ ]
```

In unserem Beispiel mit den Logiken Eins, Zwei und Drei, welche bei Drücken der entsprechenden Taste die Umstimmungen E, Z bzw. D durchführen wird das Periodenintervall verändert. Bei einer solchen Veränderung müssen alle liegenden Tasten, außer Tasten im FT-Bereich, neu gestimmt werden. Da der FT-Bereich hier nur eine Taste (69) umfasst, bleibt nur diese Taste (=a') von den Umstimmungen unberührt.

6.2.2. Verändern der Verankerungstaste

Die allgemeine Form einer Veränderung der Verankerungstaste lautet

```
UMSTIMMUNG
    Name = NeuerWert [ ]
```

Wenn Sie die Verankerungstaste neu setzen wollen, z. B. auf die Taste 57, so lautet die Umstimmung hierzu:

```
UMSTIMMUNG
    Anker_Neu = 57 [ ]
```

[3] Beispielprogramm periode.mut

Wenn nun in einer Logik die Umstimmung `Anker_Neu` aufgerufen wird, so wird die Verankerungstaste auf 57 gesetzt, alle anderen Tonsystemparameter behalten ihren alten Wert. Obwohl scheinbar nur ein Parameter verändert wird, handelt es sich hierbei um den komplexesten Umstimmungstyp. Eine Veränderung des Ankers impliziert nämlich eine Veränderung fast aller Töne der FT, da nicht nur die Verankerungstaste neu gesetzt wird, sondern die komplette Intervallstruktur des alten Tonsystems auf die neue Verankerungstaste verschoben wird. Hier wird also die Struktur des alten Tonsystems auf einen neuen Bezugston kopiert, welcher dann zum neuen Anker (oder Zentrum) wird.

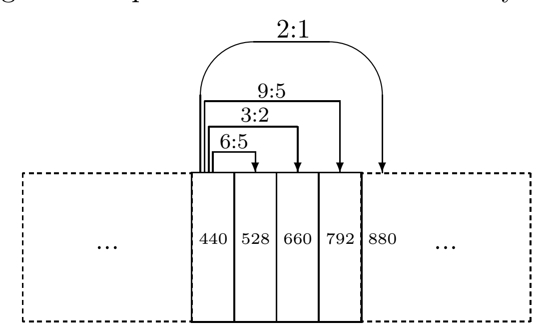
Eine Veränderung der Verankerungstaste geht folgendermaßen vonstatten:

- Der Computer merkt sich die Intervallstruktur des aktuellen Tonsystems, also das Intervall zwischen 1. und 2. Ton der FT, zwischen 1. und 3. Ton der FT. usw...
- Der Parameter 'Verankerungstaste' bekommt den in der Umstimmung angegebenen neuen Wert zugewiesen.
- Die Frequenz, die im alten Tonsystem ertönen würde, wenn die neue Verankerungstaste gedrückt würde, wird als Frequenz des 1. Tons der neuen FT eingetragen.
- Die Intervallstruktur des alten Tonsystems, die der Rechner sich gemerkt hat, bestimmt nun die Töne der neuen FT, unter Bezugnahme auf den neuen Grundton (=1. Ton) der FT. Somit erhält z. B. der 6. Ton der neuen FT die Frequenz, welche sich berechnet, wenn man die Frequenz der neuen Verankerungstaste mit dem Intervallfaktor multipliziert, den sich der Rechner als Intervall zwischen dem 1. und dem 6. Ton der alten FT gemerkt hat.

Am besten lässt sich dieser Vorgang an einem Beispiel erklären. Das Laufzeitmodul kann in einem Protokollfenster sowohl Tonsystemdaten als auch die Intervallrelationen zwischen den Tönen der Fundamentaltonleiter anzeigen. Man kann diesen Vorgang also am Bildschirm sehr gut verfolgen und nachvollziehen. Nehmen wir an, das aktuelle Tonsystem sei

```
INTERVALL
    TerzGross = 5 : 4
    TerzKlein = 6 : 5
    Oktave    = 2 : 1
TON
    i = 440
    j = i + TerzKlein
    k = j + TerzGross
    l = k + TerzKlein
TONSYSTEM
    Meier = 60 [i,j,k,l] Oktave
```

Die Abbildung zeigt die Frequenzen der Töne dieses Tonsystems:

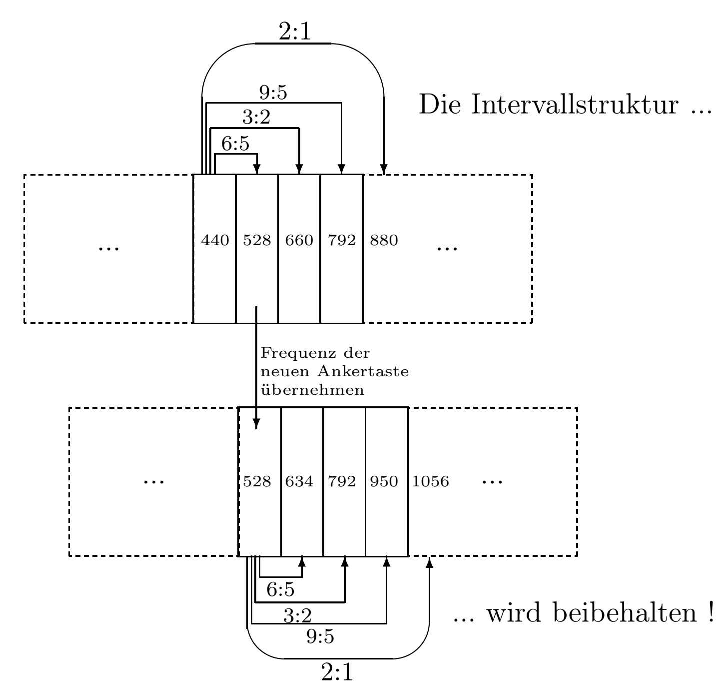


Sie können anhand dieser Grafik auch die dem Tonsystem zugrunde liegende Intervallstruktur erkennen. Ausgehend von der Verankerungstaste, welche mit einer Frequenz von 440 Hz belegt ist, gehen die drei Intervalle $\frac{6}{5}$, $\frac{3}{2}$ und $\frac{9}{5}$ aus. Deshalb lauten die Frequenzen der vier Töne der Fundamentaltonleiter 440 Hz, 528 Hz, 660 Hz und 792 Hz. Mit Hilfe des Periodenintervalls Oktave (`2:1`) ergeben sich die Frequenzen aller übrigen Tasten.

Nun soll die Verankerungstaste von ihrem alten Wert (= 60) auf den neuen Wert 61 angehoben werden, also[4]

```
UMSTIMMUNG
    Schmitt = 61 [ ]
```

Wie sieht nun das neue, durch die Umstimmung „Schmitt“ entstandene Tonsystem aus?



[4]Das komplette Beispielprogramm zur Veränderung der Verankerungstaste heißt anker.mut

Die Intervallstruktur wurde aus dem alten Tonsystem auf den neuen Ankerton übertragen. Dieses Prinzip liegt allen Veränderungen des Verankerungstones zugrunde.

Um die Intervallstruktur des Tonsystems bei einer Änderung der Verankerungstaste beizubehalten, ist es also meist notwendig, auch die Frequenzzuordnungen der Töne der Fundamentaltonleiter zu verändern. Wenn Sie eine solche Umstimmung durchführen, während eine oder mehrere Töne liegen, so muss deren Frequenz korrigiert werden[5]

Doch hier noch eine Randbemerkung: Bei Tonsystemen der Breite 1 hat eine Veränderung der Verankerungstaste keine hörbare Auswirkung. (Man überlege sich, warum!)

6.2.3. Verändern der Breite der FT

Eine recht seltene Umstimmungsart ist die Veränderung der Breite der Fundamentaltonleiter. Sie können mit diesem Umstimmungstyp die Breite der FT auf einen beliebigen neuen Wert setzen. Es gibt deshalb drei verschiedene Fälle:

- Der erste Fall ist trivial: die neue Breite entspricht der alten Breite – damit wird gar nichts verändert.
- Wenn die neue Breite kleiner ist als die aktuelle Breite, so wird das Periodenintervall neu gesetzt, und zwar auf den Wert des Intervalls zwischen dem ersten Ton der FT und der ersten Taste nach der neuen Breite. Dann werden alle Töne der FT vergessen, die jenseits der neuen Breite lagen.
- Wenn die neue Breite größer ist als die aktuelle Breite, so wird ebenfalls das Periodenintervall auf das Intervall zwischen dem ersten Ton der FT und dem ersten Ton nach der neuen Breite gesetzt. Dann werden die Töne, um die die FT erweitert wurde, gemäß dem alten Tonsystem berechnet und eingetragen.

Man kann also sagen, dass eine Veränderung der Breite der FT im allgemeinen auch eine Veränderung der Töne der FT und des Periodenintervalls nach sich zieht.

Syntaktisch sieht diese Konstruktion so aus:

```
UMSTIMMUNG
  Neue_Breite = [ << 7 >> ]
```

(Die doppelten Kleiner-als- bzw. Größer-als-Zeichen (« bzw. » sollen an eine Breitenangabe in Zeichnungen, wie z. B. $\longleftarrow 4cm \longrightarrow$ erinnern.) In diesem Beispiel wurde die Breite der FT auf den neuen Wert 7 gesetzt.

Eine praktische Bedeutung der Breitenänderung ist das *Expandieren* von Tonsystemen zur Benutzung von Harmonie-Auslösern, die eine größere Breite der FT voraussetzen. Da der Begriff des Harmonieauslösers noch nicht erklärt wurde, sei

[5] Siehe dazu auch Abschnitt „Korrektur der Frequenzen“

hier nur soviel bemerkt, dass gewisse Konstruktionen eine FT der Breite 12 verlangen. Wenn Sie ein gleichstufig temperiertes Halbtonsystem programmieren, das die Breite 12 hat, so würden Sie bisher

```
INTERVALL
  Halbton = 12 Wurzel 2
  Oktave  = 2 : 1

TON
  c   = a - 9 Halbton       cis = a - 8 Halbton
  d   = a - 7 Halbton       dis = a - 6 Halbton
  e   = a - 5 Halbton       f   = a - 4 Halbton
  fis = a - 3 Halbton       g   = a - 2 Halbton
  gis = a - 1 Halbton       a   = 440
  ais = a + 1 Halbton       h   = a + 2 Halbton

TONSYSTEM
  Halb12 = 60 [ c,cis,d,dis,e,f,fis,g,gis,a,ais,h ] Oktave
```

schreiben müssen.

Eine starke Vereinfachung dieser Stimmung stellt nun eine auf 12 expandierte FT der (alten) Breite 1 dar. Programmieren Sie einfach ein Halbtonsystem mit der Breite 1, wie Sie es im Beispiel auf Seite 33 kennen gelernt haben. Wenn Sie nun die Breite der FT auf 12 setzen, das Tonsystem also expandieren, so stellt sich keinerlei hörbarer Effekt ein, da das Periodenintervall dieser Änderung angeglichen wird[6]. Auf diese Weise haben Sie ohne Mühe ein gleichstufiges Halbtonsystem der Breite 12 programmiert:

```
INTERVALL
  Halbton = 12 Wurzel 2

TON
  a = 440

TONSYSTEM
  Halb1 = 69 [ a ] Halbton

UMSTIMMUNG
  Expandieren = [ << 12 >> ]
```

Wenn eine Logik die Umstimmung `Expandieren` durchführt, während das Tonsystem `Halb1` aktiv ist, so wird der gewünschte Zustand eines gleichstufigen Halbton-

[6]Eine solche Expansion funktioniert auf nichtveränderunde Weise nur bei einer Vergrößerung um ein ganzzahliges Vielfaches der alten Breite.

systems mit der Breite 12 eingestellt[7].

6.2.4. Verändern der Töne der FT

Dies ist die wohl vielfältigste Umstimmungsart. Während die bisher erklärten Umstimmungstypen die Töne der FT zumindest in ihrem intervallischen Zusammenhang nicht manipuliert haben, so können Sie mit diesem Umstimmungstyp die Struktur des aktuellen Tonsystems gezielt verändern. Ein weiteres Beispiel: Nehmen wir an, Sie spielten gerade das

```
TONSYSTEM    C_Dur = 60 [ c,des,d,es,e,f,fis,g,as,a,b,h ] Oktave
```

und wollen die Töne des,es und as in cis,dis und gis umwandeln (Wir nehmen der Einfachheit halber an, dass alle diese Töne im Tondeklarationsteil bereits deklariert worden sind). Die Umstimmung hierzu lautet

```
UMSTIMMUNG
    Be = [@,cis,@,dis,@,@,@,@,gis,@,@,@]
```

Sie werden sich nun fragen, was das „Klammeraffensymbol" hier zu suchen hat. Hätte man eine solche Umstimmung nicht einfacher

```
    Be = [,cis,,dis,,,,,gis,,,]
```

schreiben können? Aber erinnern wir uns, was ein Leerschritt zwischen den Kommata bei Tonsystemen bedeutet: die Taste wird „gesperrt". Und wenn wir hier den Klammeraffen weggelassen hätten, wäre auch genau das passiert. Alle Tasten, außer cis,dis und gis wären gesperrt worden (Durch die Anwendung von Leerstellen können Sie gezielt in einer Umstimmung bestimmte Tasten sperren). Die Benutzung des Klammeraffen hingegen bedeutet: „*alter Wert*", d. h. es wird der Wert genommen, den der Parameter vor der Umstimmung hatte. Wenn Sie in einer Umstimmung nur den Klammeraffen benutzen, so wird nichts verändert. Der Klammeraffe kann auch bei allen anderen Umstimmungsarten benutzt werden:

```
UMSTIMMUNG
    Otto = @ [ ]
    Meier = [<<@>>]
    Fritz = [ ] @
    Schorsch = [@,@,@,@,@,@,@,@,@,@,@,@]
```

All diese Umstimmungen bewirken nichts und haben für die Praxis sicherlich keine Bedeutung, aber sie führen uns zur zweiten Kategorie von Umstimmungen, den *relativen Umstimmungen*. Hier wird das Klammeraffensymbol (`@`) gebraucht, um in einer Formel zur Bestimmung des neuen Wertes auf den „alten Wert" zugreifen zu können.

[7] Später lernen Sie ein Verfahren kennen, mit dem man diese beiden Schritte in einer einzigen Umstimmung zusammenfassen kann (Umstimmungsbund).

6.3. Relative Umstimmungen

Die Funktionsweise der relativen Umstimmungen ist die gleiche wie bei den absoluten. Der einzige Unterschied besteht darin, dass dem umzustimmenden Parameter kein fester Wert zugewiesen wird, sondern dass sich der neue Wert aus dem alten Zustand mit einer einfachen Formel berechnen lässt.

Dies ist nützlich, wenn der neue gewünschte Zustand eines Parameters nicht absolut bekannt ist, sondern sich aus dem alten Zustand ergibt. Dies ist der Fall, wenn z. B. die Verankerungstaste um eins erhöht werden soll. Dies schreibt man so:

```
UMSTIMMUNG
    Anker = @ + 1 [ ]
```

Das Klammeraffensymbol (`@`) bedeutet hier 'alter Wert'. Bei dieser Umstimmung wird die Verankerungstaste auf 'alter Wert plus eins' gesetzt, sprich um eins erhöht. Wenn die Verankerungstaste zum Zeitpunkt des Umstimmungsaufrufs den Wert 56 hat, so ist ein Aufruf der Umstimmung `Anker` gleichbedeutend mit der (absoluten) Umstimmung

```
UMSTIMMUNG
    Von_56 = 57 [ ]
```

Relative Umstimmungen können genauso wie absolute Umstimmungen jeden der Grundparameter Anker, Breite und Töne des aktuellen Tonsystems sowie das Periodenintervall verändern. Die nachfolgend aufgeführten Umstimmungen sind alle gültig:

```
UMSTIMMUNG
    Anker  = @ + 1 [ ]
    Breite = [ << @ - 2 >> ]
    Toene  = [@ , @ , @ + 2 Quinte - Oktave ,
              @ - 3 Terz + Oktave , @ , @ + Syn_Komma , , ]
    Periode= [ ] @ + Syn_Komma
```

An dieser Stelle sei bemerkt, dass man als Namen keines der *reservierten Worte* verwenden darf. Dazu zählt das Wort `Ton`. Wenn man beispielsweise diese Umstimmungen deklariert,

```
UMSTIMMUNG
    Anker  = @ + 1 [ ]
    Breite = [ << @ - 2 >> ]
    Ton  = [ @ , @ , @ + 2 Quinte - Oktave ,
             @ - 3 Terz + Oktave , @ , @ + Syn_Komma , , ]
    Periode= [ ] @ + Syn_Komma
```

so erhält man einen Syntaxfehler, weil das Wort `Ton` eine vorgegebene Bedeutung innerhalb der Programmiersprache hat. Erlaubt hingegen ist, wenn ein reserviertes

Wort Bestandteil eines Bezeichners ist, so ist es zwar verboten, eine Umstimmung namens `Ton` zu deklarieren, aber eine Umstimmung `Tontaube` ist natürlich erlaubt.

Da die vier verschiedenen Umstimmtypen in ihrer Bedeutung und Handhabung sehr unterschiedlich sind, folgt hier noch einmal eine Auflistung der gültigen Rechenzeichen bei relativen Umstimmungen:

Parameter	gültige Rechenzeichen
Anker	+ -
Breite	+ - * /
Periode	+ -
Töne der FT	wie bei TON-Deklaration

Bereichsüber- bzw. Unterschreitungen werden im Laufzeitmodul abgefangen und ignoriert. Wenn z. B. die Breite der FT 1 beträgt und die Umstimmung

```
UMSTIMMUNG
    Halbieren = [ << @/2 >> ]
```

aufgerufen wird, so wird die angegebene Umstimmung ignoriert, da die Breite der FT mindestens 1 betragen muss. Der Anschauung halber folgt nun noch einmal eine Liste gültiger relativer Umstimmungen:

```
INTERVALL
  cent = 1200 wurzel 2

UMSTIMMUNG
  Verdopplung    = [ << @ * 2 >> ]
  Trichter       = [ ] @ + 3 cent
  Temp_nach_rein = [ @ , @ , @ , @ , @ - 14 cent , @ - 2 cent ,
                     @ , @ + 2 cent ]
  Transponieren  = @ - 2 [ ]
  Was_ist_das    = @ + 4
```

Die letzte Umstimmung ist syntaktisch falsch, da nicht eindeutig ist, ob die Verankerungstaste oder das Periodenintervall verändert werden soll. Aus diesem Grunde ist es immer notwendig, die zunächst als überflüssig erscheinenden eckigen Klammern zu schreiben.

Detailliertere Informationen über Umstimmungen mit einer exakten Spezifikation der Vorgehensweise in Ausnahmesituationen entnehmen Sie bitte dem Abschnitt „Handhabung von Grenzfällen“.

6.4. Umstimmungsbünde

Mit den bisher gezeigten Umstimmungsarten können Sie nur jeweils *einen* Parameter verändern. Manchmal ist es jedoch erforderlich, zwei oder drei Parameter in einem

Umstimmvorgang zu ändern. Es wäre nun prinzipiell denkbar, einfach die Syntax der Tonsystemdeklaration zu übernehmen, um eine gleichzeitige Veränderung von Breite und Periodenintervall zu erreichen, also etwa

```
UMSTIMMUNG
    Breite_und_Periode = [ << @+9 >> ] Quint
```

Leider ist eine solche Konstruktion in ihrem Ergebnis nicht eindeutig. Es macht einen deutlichen Unterschied, ob zuerst das Periodenintervall verändert wird und dann die Breite, oder umgekehrt[8]. Sie können deshalb in einem Umstimmungsbund mehrere Umstimmungen nacheinander auflisten, um so eine eindeutige Reihenfolge festzulegen. Obiges Beispiel würde demnach wie folgt umgeschrieben werden müssen:

```
UMSTIMMUNG
    Breite   = [ << @+9 >> ]
    Periode  = [ ] Quint
    Entweder = { Breite, Periode }
    Oder     = { Periode, Breite }
```

Der Umstimmungsbund `Entweder` vergrößert zuerst die Breite des momentanen Tonsystems um neun, um dann das Periodenintervall auf Quinte zu setzen; der Umstimmungsbund `Oder` führt die gleichen Umstimmungen in umgekehrter Reihenfolge durch.

In einem Umstimmungsbund können Umstimmungen, Tonsysteme und sogar Logiken[9] und andere Umstimmungsbünde[10] aufgelistet werden. Alle aufgelisteten Elemente eines Umstimmungsbundes müssen im Programm deklariert sein.

[8] überlegen Sie sich, wie diese Umstimmung unterschiedlich auf das Periodenintervall eines Tonsystems der Breite 1 wirkt (z. B. das Tonsystem Drittel).
a) zuerst wird die Breite auf den Wert 10 erhöht, dabei ändert sich das Periodenintervall, was jedoch egal ist, da es anschließend auf den absoluten Wert Quinte gesetzt wird.
b) Wird hingegen zuerst das Periodenintervall auf Quinte gesetzt und anschließend die Breite um 9 Tasten (also auf den Wert 10) expandiert, so wird das Periodenintervall dieser Breitenänderung angepasst und erhält einen ungemein größeren Wert.

[9] Bei Namensmehrdeutigkeiten wird hier erstrangig zugunsten der Logik, dann zugunsten der Umstimmung entschieden !

[10] Hierbei ist zu beachten, dass keine Rekursionen entstehen dürfen.

7. Logiken

Kommen wir nun zur „Schaltzentrale“ unserer Stimmung - der Logik. Hier bestimmen Sie, welche Ereignisse Umstimmungen auslösen sollen. Die Logik ist außerdem das einzige Objekt aus Ihrem Programm, das die Verbindung zur Benutzeroberfläche des Laufzeitmoduls herstellt. Alle anderen Objekte wie Intervalle, Tonsysteme, … sind rein „deklarative Elemente“, die erst dann im Laufzeitmodul eine Wirkung zeigen, wenn sie von einer Logik benutzt werden. Aus diesem Grund mussten Sie bisher, selbst wenn Sie nur ein einfaches Tonsystem spielen wollten, dem Programm eine Logik der Gestalt

```
LOGIK
    Name Taste x = Name [ ]
```

hinzufügen. Die Deklaration von Tonsystemen hat im Laufzeitmodul noch keine Auswirkung. Nur Elemente, die direkt oder indirekt von einer Logik benutzt werden, werden in das Laufzeitmodul eingetragen.

Eine Logik besteht aus drei Teilen: dem *Auslöser*, der *Einstimmung* und dem *Aktionsteil*. Die allgemeine Struktur einer Logik lautet also:

```
LOGIK Name Auslöser = Einstimmung [ Aktionen ]
```

7.1. Der Auslöser

Um eine Stimmungslogik zu aktivieren, ist irgend eine Art von externem Signal nötig. Dies kann z. B. ein Tastendruck auf der Computertastatur sein. Wenn z. B. die Stimmungslogik `Meier` durch das Drücken der Taste 'M' aktiviert werden soll, so schreiben Sie

```
LOGIK
      Meier Taste M = ...
```

Mutabor kennt noch weitere Auslöser für Stimmungslogiken, die aber erst im Teil III dieses Handbuches beschrieben werden.

7.2. Die Einstimmung

Hier kann ein Tonsystemname oder ein Umstimmungsname stehen. Das angeführte Tonsystem (bzw. die Umstimmung) wird einmalig aufgerufen, und zwar in dem Moment, in dem Sie im Laufzeitmodul die Logik durch Tastendruck[1] aktivieren.

[1]bzw. durch einen anderen Auslöser

Eine Einstimmung dient als Initialisierung des aktuellen Tonsystems. Wenn Sie hier ein Tonsystem eintragen, spricht man von einer „harten“ Initialisierung, weil der alte Zustand komplett vergessen wird und alle Parameter auf einen neuen Wert eingestellt werden, gemäß dem Einstimmungs-Tonsystem. Tonsysteme werden immer dann als Einstimmung benutzt, wenn die Logik einen fest definierten und somit immer gleichen Anfangszustand haben soll. Und natürlich, wenn Sie, wie im Abschnitt „Tonsysteme“, nur ein nicht-mutierendes Tonsystem spielen wollen. In diesem Fall besteht die Logik aus einer Einstimmung (nämlich dem gewünschten Tonsystem) und einer leeren Aktionsliste („ `[ ]` “).

Wenn Sie eine Umstimmung als Einstimmung angeben, so wird nur ein Teil der Parameter verändert. Solche „weichen“ Einstimmungen sind mit Vorsicht zu genießen, da verschiedene Ausgangszustände die Logik unterschiedlich initialisieren. So passieren möglicherweise ungewollte Effekte. Nichtsdestotrotz sind Umstimmungen als Einstimmung einer Logik manchmal nützlich, insbesondere wenn nur die Breite des Tonsystems expandiert werden soll.

Außerdem ist es möglich, keine Einstimmung anzugeben. Dann wird das aktuelle Tonsystem beim Aufruf der Logik nicht verändert, es findet beim Aufruf der Logik überhaupt keine Initialisierung statt.

7.2.1. Eine Randbemerkung

Sie können jetzt eine „leere“ Logik programmieren:

```
LOGIK
    Leer Taste L = [ ]
```

Der Sinn einer solchen Logik ist folgender: nehmen wir an, Sie hätten eine mutierende Stimmung programmiert, auf ihr gespielt und einen Punkt erreicht (sprich: eine Stimmung), den Sie gerne unverändert weiterspielen wollen. Leider würde die Automatik der mutierenden Stimmung diesen gewünschten Zustand bald wieder verlassen.

Wenn Sie nun eine „leere“ Logik aktivieren, so wird, da sie keine Einstimmung hat, eben der aktuelle gewünschte Zustand beibehalten; und, da in der leeren Logik keine Aktionen definiert sind, dieser Zustand auch nicht verändert.

Doch nun wird es Zeit, von solchen statischen Logiken zu den mutierenden, automatischen Logiken zu kommen.

7.3. Harmonien als Auslöser für Umstimmungen

Innerhalb der eckigen Klammern einer Stimmungslogik können Sie nun beliebig viele Anweisungen der Gestalt

```
Auslöser -> Aktion
```

angeben.

In den meisten Fällen wird die Aktion eine Umstimmung sein, die das aktuelle Tonsystem verändert. Die Auslöser regeln nun eindeutig, unter welchen Bedingungen die Umstimmungen durchgeführt werden sollen.

Am einfachsten können Sie innerhalb einer Stimmungslogik Umstimmungen per Tastendruck erzeugen. Wenn Sie z. B. den Unterschied zwischen einer Naturseptime (7:4) und der Septime als Schichtung zweier Quarten (16:9) unmittelbar vergleichen möchten, so ist es sinnvoll, diese Umstimmungen mit einer Taste als Auslöser zu versehen, also z. B.[2]:

```
LOGIK
      Reines_C_Dur Taste R = C_Dur [
                                     Taste N -> Natursept
                                     Taste S -> Normalsept
      ]
UMSTIMMUNG
      Natursept  = [ @,@,@,@,@,@,@,@,@,@,@-Septkomma,@ ]
      Normalsept = [ @,@,@,@,@,@,@,@,@,@,@+Septkomma,@ ]
INTERVALL
      Septkomma = 64:63
```

Das Tonsystem `C_Dur` ist bereits des öfteren weiter oben benutzt worden (Siehe Seite 33). Versuchen Sie diese Stimmungslogik einmal, schlagen Sie einen vollen C^7-Akkord (c',e',g',b') an und wechseln Sie per Tastendruck (N bzw. S) zwischen den beiden Septimen – ein sehr interessanter Vergleich!

Dieses einfache Prinzip zum Auslösen von Umstimmungen bietet zwar bereits eine Reihe an Möglichkeiten, ist aber dennoch recht unflexibel. Das Hinzukommen eines jeden neuen Bedienungselementes ist für den Spieler ungewohnt und fremd. Wer möchte schon gerne während des Spiels ständig einen Knopf am Computer drücken, um die rechte Intonation zu bekommen.

Deshalb besitzt Mutabor die Fähigkeit zu einer einfachen programmierbaren harmonischen Analyse des gespielten Musikstücks. Sie können Umstimmungen jetzt von gespielten Harmonien abhängig machen. Der Spieler braucht sich an keine neuen Knöpfe zu gewöhnen – die Umstimmung passiert *automatisch*.

Wo liegen nun die praktischen Anwendungen einer solchen harmoniegekoppelten Umstimmungsautomatik? Nehmen wir an, Sie wollen Stücke in tonaler Dur-Moll-Harmonik spielen, aber mit rein gestimmten Terzen und Quinten. Zu diesem Zweck können Sie eine sehr einfache Stimmungslogik programmieren, die Terzen, Quinten und Dreiklänge als solche erkennt und die passende Umstimmung durchführt, so dass diese Intervalle rein intoniert werden können[3].

7.3.1. Die Projektionstonleiter

Um die Funktionsweise von mutierenden Stimmungslogiken verstehen zu können, müssen wir den Begriff der Projektionstonleiter einführen.

Die gesamte Klaviatur lässt sich als blockweise Aneinanderreihung von Tonleitern verstehen. Eine besondere Tonleiter ist die Fundamentaltonleiter. In ihr stecken die

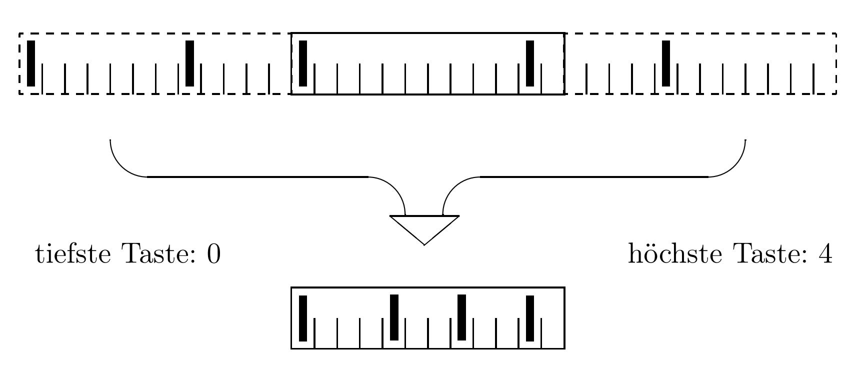
[2]Beispielprogramm naturspt.mut

[3]Eine solche Stimmungslogik finden Sie im Abschnitt 11.2.

Informationen der vier Grundparameter: Töne, Breite, Anker und Periode. Bedenken Sie, dass trotz der 12-periodischen Struktur der Klaviatur die Fundamentaltonleiter – und somit alle momentanen „Tonleitern“ nicht auf eine Breite von 12 Tasten festgelegt sind! Somit kann es möglich sein, dass die Klaviatur bei einem Tonsystem der Breite 7 vom c' zum g' zum d" … in Blöcke unterschiedlicher Tastenstruktur unterteilt ist!

Die Projektionstonleiter enthält nun keine Frequenz- oder Tonwerte, sondern die Information, welche Tasten der Klaviatur momentan gedrückt sind, und zwar *in die FT projiziert*. Ein Beispiel: Ihre FT ist auf Taste c' (=60) verankert und habe die Breite 12, also eine ganz normale C-Dur-Tonleiter. Bezeichnen wir den Zustand „Taste gedrückt“ mit '*' und den Zustand „Taste nicht gedrückt“ mit '+'. Solange wir keine Taste auf der Klaviatur anschlagen, hat die Projektionstonleiter das Aussehen „++++++++++++“. Wenn wir jetzt die Taste c' (=60) anschlagen, so ändert sich der Zustand der Projektionstonleiter zu „*+++++++++++“, da die erste Taste einer Tonleiter gedrückt ist. Wenn wir einen C-Dur-Akkord anschlagen, (Tasten c', g' und e'), so enthält die PT das Muster „*+++*++*++++“.

Warum heißt dieses Gebilde Projektionstonleiter? Weil es ohne Bedeutung ist, in welcher Tonleiter (oder Lage) eine Taste angeschlagen wird, alles wird in die FT hineinprojiziert. Musikalisch gesprochen ist die Projektionstonleiter also nichts anderes als die Menge der momentan gedrückten *Tonigkeiten*. So ist der Zustand der PT gleich „++*+++*++*++“, wenn man die Tasten d'(=62),fis'(=66) und a'(=69) anschlägt, und ebenso, wenn man die Tasten d (=50),a' (=69) und fis“ (=78) drückt. Bei einem zwölfstufigen Tonsystem gibt es also – laut Kombinatorik – (theoretisch) 4096 verschiedene mögliche Zustände der PT (inklusive keiner gedrückten Taste), allgemein gibt es 2^{Breite} mögliche Zustände einer PT.



Die Projektionstonleiter enthält
die Harmonie { 0,4,7,10 }

Ein zweites, etwas seltsames Beispiel: Betrachten wir ein Tonsystem, wie wir es für die Einstimmung von gleichstufig temperierten Tonsystemen benutzt haben. Setzen wir den Anker auf die Taste a' (=69) und die Breite der FT auf 1 (Die Töne und des Periodenintervalls sind für diese Betrachtung nicht von Bedeutung). Überlegen wir uns einmal, welche verschiedenen Zustände die PT haben kann. Ein seltsames Ergebnis: nur zwei. Wenn keine Taste auf der gesamten Klaviatur gedrückt ist, so

bekommt die PT den Inhalt „+“. Und sobald irgend eine oder mehrere Tasten gedrückt werden bekommt die PT den Inhalt „*“. Warum? Nun, weil alles in die FT projiziert wird, und die FT hat nur die Breite eins.

Fassen wir zusammen: *die PT enthält die gleiche Anzahl „Tasten“, wie die FT breit ist. In ihr wird der in den Bereich der FT projizierte Zustand der Klaviatur festgehalten.*

7.4. Harmoniedeklarationen

Nach dieser etwas komplizierten Einführung des Begriffes der Projektionstonleiter kommen wir nun zu dessen praktischer Anwendung, der *Harmoniedeklaration.* Am einfachsten lässt sich dies an einigen Beispielen verdeutlichen.

Nehmen wir an, wir befänden uns in einem zwölfstufigen Tonsystem mit Verankerungstaste c’ (=60). Wir wollen einen C-Dur-Akkord (Tasten c,e und g) als Auslöser für eine Umstimmung benutzen. Diese Harmonie (`C_DUR`) deklarieren wir als:

```
HARMONIE
    C_DUR = { 0,4,7 }
```

Die Harmonie `C_DUR` wird erkannt, sobald die 0., die 4. und die 7. Taste der Projektionstonleiter gedrückt sind. Bitte führen Sie sich vor Augen, dass der Begriff „Harmonie“ in diesem Zusammenhang lediglich ein bestimmtes Tastenmuster auf der Klaviatur meint.

Neben den aufgeführten Tasten einer Harmonie, die gedrückt sein *müssen*, können Sie auch Tasten angeben, deren Zustand bei der Harmonieanalyse *nicht* berücksichtigt werden soll:

```
HARMONIE
    Dur2 = { 0,4,7,*10 }
```

Der Zustand einer mit einem Stern versehenen Taste wird bei der Harmonieanalyse ignoriert. Die Harmonie `Dur2` ist also erfüllt, sowohl beim Spielen der Harmonie `{0,4,7,10}` , als auch `{ 0,4,7 }` .

7.4.1. Eine Anmerkung

Die hier vorgestellten Prinzipien einer Harmonieanalyse entstammen – wie unschwer zu erkennen ist – den Begriffen einer einfachen dur-moll-tonalen Harmonik. Wir denken jedoch, dass diese Harmonieanalyse von Mutabor vielfältigere Verwendung finden kann, da hier praktisch alle möglichen Kombinationen liegender Tasten eine Aktion (z. B. Umstimmung) veranlassen können.

Oft taucht der Wunsch auf, Umstimmungen nicht nur von harmonischen – und somit momentanen – Ereignissen, sondern auch von *melodischen* Prinzipien auslösen zu können. Leider stellt dies zwei gravierende Probleme prinzipieller Art: einerseits

ist die Live-Verwendbarkeit eine wichtige Fähigkeit von MUTABOR. Ob eine Umstimmung oder allgemeiner eine Aktion stattfinden soll, oder nicht, muss sofort, also nur unter Berücksichtigung vergangener und aktueller, nicht aber zukünftiger Ereignisse entschieden werden können. Melodische Strukturen benötigen leider meistens Entscheidungsstrukturen der Art „vor zwei Minuten hätte eine Umstimmung stattfinden sollen". Das zweite Problem bei melodischen Auslösern ist, dass sie nur schwer in allgemeiner Weise als Programmiersprache formuliert werden können. Sie können jedoch sicher sein, dass wir an diesem Problem arbeiten. Für Ideen, Vorschläge oder Anregungen sind wir jederzeit dankbar.

Obwohl den meisten Beispielen ein zwölfstufiges Tonsystem zugrunde liegt, können Sie praktisch bei jeder beliebigen Breite für ein Tonsystem Harmonieerkennungen verwenden. Leider sind wegen der Struktur der Klaviatur z. B. 7-stufige Tonsysteme denkbar ungünstig zu spielen, so dass man meistens bestimmte Tasten stumm schaltet, wie in der Beispiellogik `PENTA2` in Abschnitt 5.5.3 gezeigt wurde. Schön wäre es natürlich, wenn es eine MIDI-Klaviatur gäbe, die nicht auf die Zwölferstruktur festgelegt ist.

7.5. Der Aktionsteil

Hier wird die Umstimmautomatik einer mutierenden Stimmung definiert. Eine Umstimmautomatik besteht aus einer Aneinanderreihung von Anweisungen. Wir wollen in diesem Abschnitt besprechen, welche Anweisungen Sie zur Programmierung einer mutierenden Stimmung verwenden können.

Wie der Begriff schon andeutet, soll es sich bei der Umstimmautomatik um einen im Hintergrund arbeitenden Mechanismus handeln, der die Stimmung des Instrumentes immer im richtigen Moment verändert. Bereits bei der Besprechung von Umstimmungen haben Sie einen solchen Mechanismus kennen gelernt: die Umstimmung 'auf Knopfdruck'. Leider ist es während des Spiels nicht immer möglich, die eine Hand zum Computer zu bewegen, um dort eine Taste zur Umstimmung zu betätigen. Deshalb können Sie MUTABOR so programmieren, dass bei der Erkennung von bestimmten Tastenkombinationen (Harmonien) eine Umstimmung durchgeführt wird. Welche Tastenkombination nun welche Umstimmung hervorrufen soll, können Sie frei bestimmen.

Es gibt insbesondere in der Anwendung tonaler Harmonik bestimmte Strukturen, die über pure „Tastenkombinationen" herausgehen. Deshalb wäre es meistens sehr umständlich, wenn Sie alle möglichen Tastenkombinationen angeben müssten, die die Umstimmung erzeugen sollen. Die Analyse der Klaviatur zu jedem Zeitpunkt einer Änderung steht in einem Zusammenhang mit dem aktuellen Tonsystem, insbesondere mit seiner Breite. Ein sinnvoller Auslöser für Umstimmungen ist also die Erkennung einer Harmonie, bzw. eine Beziehung zwischen den gedrückten Tasten und der Fundamentaltonleiter, was dann z. B. so etwas wie einen „C-Dur-Akkord ohne Berücksichtigung von Lage und Umkehrung" ergibt. Allgemein können wir also wieder

`Auslöser -> Aufruf`
als Regel für das Laufzeitmodul definieren, beim Erkennen eines bestimmten Auslösers (z. B. einer Harmonie) den angegebene Aufruf durchzuführen (z. B. eine Umstimmung aufrufen). Damit können wir bereits ein erstes Programm mit einer automatisch mutierenden Stimmung schreiben.

Nehmen wir folgendes Beispiel: Das im Abschnitt „Tonsysteme“ beschriebene Programm zum Spielen in einem 'reinen' C-Dur soll die Grundlage für unsere mutierende Stimmung sein. Dieses Tonsystem definiert den Ton b als Quinte unter dem Ton f. Nehmen wir nun an, wir hätten den Wunsch, beim Anschlagen der Sept c-b das b nicht als Quinte unter f zu intonieren, sondern als Naturseptime über c. Ferner soll bei einstimmigem Melodiespiel das b wieder als Quinte unter f benutzt werden.

Die Automatik, die dies realisieren kann, benötigt also zwei verschiedene Zustände, nämlich den Zustand „Natur“, in dem das Intervall c-b eine Naturseptime ist und den Zustand „Normal“, in dem der Ton b als Quinte unter f genommen wird. Die einfachste Realisierung dieser zwei Zustände wären zwei Logiken, bei denen diese Zustände als Einstimmung angegeben sind. Dieses Beispiel funktioniert ohne Klaviaturanalyse. Im Programmtext also:

```
INTERVALL
    Quinte       = 3:2
    Terz         = 5:4
    Naturseptime = 7:4
    Oktave       = 2:1
TON
    c   = a - Terz + Quinte - Oktave
    des = f - Terz
    d   = g + Quinte - Oktave
    es  = g - Terz
    e   = c + Terz
    f   = c - Quinte + Oktave
    fis = d + Terz
    g   = c + Quinte
    as  = c - Terz + Oktave
    a   = 440
    b   = f - Quinte + Oktave
    h   = g + Terz
    b2  = c + Naturseptime
TONSYSTEM
    C_Dur_Rein = 60 [ c,des,d,es,e,f,fis,g,as,a,b,h ] Oktave
UMSTIMMUNG
    Natur =  [ @,@,@,@,@,@,@,@,@,@, b2 ,@ ]
LOGIK
    C_Normal Taste C = C_Dur_Rein [ ]
    C_Natur  Taste N = Natur [ ]
```

In diesem Programm wird zunächst ein Tonsystem mit einem 'reinen' C-Dur programmiert, wie Sie es aus dem Kapitel „Tonsysteme“ bereits kennen. Zusätzlich wird das Intervall Naturseptime als `7:4` und der Ton `b2` als `c+Naturseptime` deklariert. Nun folgt eine Umstimmung: die Umstimmung `Natur` setzt die Taste b auf den Ton `b2`, also eine Naturseptime über c. Alle anderen Töne der FT bleiben bei der Umstimmung unverändert (Klammeraffensymbol). Zum Umschalten wurden hier einfach zwei aktionslose Logiken deklariert, die die beiden Zustände `Natur` und `C_Dur_Rein` als Einstimmung benutzen. Sie können jetzt per Knopfdruck zwischen den beiden Septimen wechseln. Wenn Sie aber gerne beide Hände zum Spielen frei haben möchten, und dennoch nicht auf den „Septimenwechsel“ verzichten wollen, so

müssen Sie sich eine Automatik programmieren. Überlegen wir uns dazu, welche Harmonien die Auslöser für Umstimmungen sind. Die Naturseptime soll in einem Zweiklang c-b erklingen. Deklarieren wir also die Harmonie

```
HARMONIE    Septakkord = { 0, *4, *7, 10 }
```

und die Aktion

```
LOGIK
    Mutierend Taste M = C_Dur_Rein [ Septakkord -> Natur ]
```

Was bedeutet dies? Die Harmoniedeklaration sollte klar sein. Ein Septakkord ist das Tastenmuster, bei dem auf jeden Fall die 0. und die 10. Taste der Projektionstonleiter gedrückt sind, eventuell die 4. und/oder die 7. Taste, aber keine anderen. Wenn der Computer während des Spiels diese Harmonie erkennt, so soll die Umstimmung `Natur` durchgeführt werden - das ist alles. Diese Logik benutzt als Einstimmung das 'normale' C-Dur und schaltet auf Naturseptime, sobald ein C-Septakkord erkannt wurde.

Die Schwäche dieses Programms liegt in der Tatsache, dass der Zustand „Naturseptime“, wenn er einmal erreicht ist, nicht mehr verlassen werden kann; aber gerade das wollen wir ja erreichen: z. B. beim einstimmigen Melodiespiel soll das b wieder als Quinte unter f interpretiert werden. Wir brauchen also noch eine zweite Aktion, die in den alten Zustand zurückschaltet, und zwar genau dann, wenn kein Septakkord mit c-b angeschlagen wird. Hierzu deklarieren wir die „Harmonie“ `Be` mit

```
HARMONIE
    Be = { 10 }
```

(Hier erkennen Sie deutlich, dass der Begriff „Harmonie“ eher im Sinne von „Tastenmuster“ zu verstehen ist.) Schließlich fügen wir der Logik die Aktion

```
LOGIK
    Mutierend Taste M = C_Dur_Rein [ Septakkord -> Natur
                                      Be -> Normal ]
```

hinzu. Damit haben wir unser Ziel erreicht. Zwecks Übersichtlichkeit hier nochmal der Steuerungsteil des Programms[4]:

[4]Beispielprogramm c_b.mut

```
UMSTIMMUNG
    Natur =  [ @,@,@,@,@,@,@,@,@,@, b2 ,@ ]
    Normal = [ @,@,@,@,@,@,@,@,@,@, b,@ ]
HARMONIE
    Septakkord = { 0, *4, *7, 10 }
    Be = { 10 }
LOGIK
    Statisch  Taste s = C_Dur_Rein [ ]
    Mutierend Taste m = C_Dur_Rein [ Septakkord -> Natur
                                     Be         -> Normal ]
```

Programmieren Sie diese Logik einmal und spielen Sie sie dann folgendermaßen:

1. Durch Drücken der Taste S aktivieren Sie die Logik `Statisch`. Schlagen Sie das Intervall f-b an. Sie hören eine reine Quart. Die Septime c-b wird aber nicht als Naturseptime intoniert.

2. Aktivieren Sie die Logik `Mutierend`. Schlagen Sie wieder das Intervall f-b an. Die Quart ist rein.

3. Drücken Sie den Ton b. Wenn Sie daraufhin den Ton c mit einem sehr sanften Anschlag betätigen, können Sie hören, wie MUTABOR den Ton b zur Naturseptime umstimmt, da die Harmonie {0,10} erkannt wurde.

4. Lassen Sie das c wieder los; sie hören nun, wie das b wieder umgestimmt wird - die Harmonie {10} wurde vom Computer erkannt.

7.5.1. Eine Randbemerkung

Bitte beachten Sie beim Programmieren von eigenen mutierenden Stimmungen, dass das Laufzeitmodul von MUTABOR die Aktionsvorschriften *nach jeder Veränderung des Klaviaturzustands* durchgeht. Welche Konsequenzen dies hat, können Sie an dem Septimenbeispiel erfahren:

Aktivieren Sie die Logik „Mutierend“ und schlagen Sie c-b an. Es erklingt die Naturseptime. Wenn Sie nun das Intervall f-b anschlagen, so erwarten Sie eine nicht reine Quart, was auch im Sinne der Umstimmautomatik liegt, da ja nur bei einem einzelnen b wieder auf die normale Stimmung zurückgeschaltet werden soll. Und hier taucht nun ein prinzipielles Problem auf. Wenn Sie das Intervall f-b so anschlagen, dass zuerst das f und dann das b erklingt, so werden Sie das b als Naturseptime über c hören. Wenn Sie dasselbe Intervall aber in der Reihenfolge b, f anschlagen, so erklingt die Quart wieder rein.

Dieser Effekt resultiert aus der Tatsache, dass MUTABOR bei jeder Änderung des Klaviaturzustandes die Aktionseinträge durchgeht und eventuell umstimmt. Wenn Sie das b vor dem f anschlagen, so haben Sie für einen kurzen Moment den Zustand 10 vorliegen - und der führt laut Umstimmautomatik zur Umstimmung „Normal“. Solange Sie sehr langsam spielen, können Sie solche Seiteneffekte leicht kontrollieren.

Wenn Sie aber mehrere Töne gleichzeitig anschlagen, so ist die Reihenfolge nicht mehr kontrollierbar. Programmieren Sie Ihre Logiken also möglichst so, dass solche als „Hauptmuster“ fehl gedeuteten „Übergangsmuster“ nicht auftreten können. (Falls Sie eine Lösung für dieses prinzipielle Problem haben, so schreiben Sie uns bitte!)

7.6. Differenziertere Harmonieanalyse

Die gewöhnliche Harmonie- oder besser: Tastaturanalyse berücksichtigt nur einen Vergleich mit der Projektionstonleiter. Haben wir z. B. ein gewöhnliches 12-stufiges Tonsystem programmiert, so erlaubt eine „Harmonieanalyse“ im Sinne tonaler Dur-Moll-Harmonik nur die Erkennung des Akkordgrundtones. Auf Lage oder Umkehrung wird keinerlei Rücksicht genommen. Ob man den C-Dur-Akkord in enger oder in weiter Lage, mit der Quinte als Grundton, mit Terzverdoppelung, oder wie auch immer anschlägt, alles fällt unter die Kategorie „Dur = { 0,4,7 }“, sofern nur die Tasten c, e und g irgendwie gedrückt sind.

Um eine differenziertere Klaviaturanalyse zu ermöglichen, stellt das Laufzeitmodul zusätzlich Informationen über die höchste und die tiefste momentan gedrückte Taste zur Verfügung. Diese Informationen können als zusätzliche Bedingungen bei der Harmonieerkennung benutzt werden.

Allerdings haben Sie keinen Zugriff auf die absolute Tastennummer, sondern auf die Nummer der entsprechenden FT-Taste, also die Tonigkeit. Wie alle anderen Tasten bei der Harmonieerkennung werden auch die beiden „Randtasten“ als in die FT projiziert angegeben. Sie können also nicht abfragen, ob die tiefste gedrückte Taste ein dis” (= 75) ist, sondern nur, ob der soundsovielte Ton des Projektionstonleiter gedrückt ist[5] Ein Beispiel:

```
HARMONIE
    D_Dur = { 2,6,9 }
UMSTIMMUNG
    Gelb = [ ] MiniOktave
LOGIK
    Braun Taste b = Zwoelf_Stufig [ D_Dur -> Gelb ]
```

Dies ist eine simple Stimmungslogik, die beim Erkennen eines D-Dur-Akkordes die Umstimmung `Gelb` durchführt und damit das Periodenintervall auf `MiniOktave` verändert. Hierbei spielen Lage und Umkehrung des Akkordes keine Rolle[6]. Hauptsache ist, dass die Tasten d, fis und a irgendwo gedrückt sind.

Wir können diese Aufrufbedingung nun einschränken, indem wir die Automatik so programmieren, dass nur bei einem D-Dur-Akkord mit dem fis als tiefstem Ton die Umstimmung stattfindet. Dies formuliert man so:

[5] Was auch bedeutend mehr Sinn ergibt

[6] Die Logik benötigt, falls Sie sie ausprobieren wollen, irgendein 12-stufiges Tonsystem mit Verankerungstaste 60 als Einstimmung, da sonst die Harmonieerkennung nicht funktioniert, siehe Abschnitt „Verändern der Breite der FT“, und außerdem eine Definition des Intervalls „MiniOktave“.

```
HARMONIE
    D_Dur = { 2,6,9 }
UMSTIMMUNG
    Gelb = [ ] MiniOktave
LOGIK
    Braun Taste b = Zwoelf_Stufig [ 6 ~ D_Dur -> Gelb ]
```

Eine 6 vor der Harmonie D_Dur bedeutet, dass die tiefste Taste 'vom Typ 6' sein muss. Die Tonigkeits-Nummer der tiefsten Taste und der Harmoniename werden durch das Tilde-Zeichen (∼) getrennt. Beachten Sie, dass im folgenden Beispiel die Umstimmung Meier nie erreicht werden kann:

```
HARMONIE Moll = {0,3,7}
UMSTIMMUNG Meier = 34 [ ]
LOGIK Nie Taste n = [ 2 ~ Moll -> Meier ]
```

(Die Harmonie Moll wird bei gedrückter 2 nicht erkannt, und eben eine gedrückte 2 ist Voraussetzung für das Aktivieren der Umstimmung Meier) Doch bleiben wir bei normalen Fällen: Wenn der D-Dur-Akkord als höchste Taste ein d haben soll, so schreibt man:

```
LOGIK
    Braun Taste b = Zwoelf_Stufig [ D_Dur ~ 2 -> Gelb ]
```

Mit Hilfe der erweiterten Harmonieanalyse ist es also möglich, nicht nur den Typ eines Akkordes, sondern auch dessen Lage und Umkehrung zu berücksichtigen.

8. Komplexe Stimmungslogiken

In diesem Teil des Handbuches zu MUTABOR haben Sie die grundlegenden Programmierkenntnisse erworben, die es Ihnen ermöglichen, beliebige statische Tonsysteme zu programmieren. Darüber hinaus können Sie einfache Harmonieerkennungen programmieren, die beliebige Parameter des momentanen Tonsystems verändern können.

Die Programmiersprache von MUTABOR enthält noch diverse weiterführende Konstruktionen, die bisher noch nicht erwähnt wurden. Alle weiteren Möglichkeiten erfahren Sie im nun folgenden Teil des Handbuches.

Es ist jedoch nicht sinnvoll, diesen Teil zu studieren, ohne bereits einige praktische Erfahrungen im Programmieren von Tonsystemen und einfachen mutierenden Stimmungen gemacht zu haben. Falls Sie also gerade dabei sind, dieses Handbuch im „Trockendurchgang“ zu studieren, so möchten wir Sie bitten, mit dem Lesen einen Moment aufzuhören und alles, was Sie bisher über das Programmieren von Stimmungslogiken gelernt haben, einmal in der Praxis auszuprobieren. Dies kostet zwar Zeit, aber Klavierspielen erlernt man ja auch nicht durch das Lesen von Noten, sondern nur durch praktisches Üben. Sie werden sehr schnell ein Gefühl für mikrotonale Strukturen entwickeln und Stimmungen nach Ihren eigenen Wünschen entwerfen – oder historische Stimmanweisungen nachvollziehen und auf Ihrer MUTABOR II einstellen und ausprobieren. Erst wenn Sie mit diesen einfachen Programmiertechniken wirklich vertraut sind, sollten Sie sich auf den nun folgenden Teil des Handbuches stürzen.

Teil III.

Fortgeschrittenes Programmieren

9. Resumée

Dieser Teil der Dokumentation zu MUTABOR befasst sich mit weiterführenden Programmiertechniken zum Erstellen komplexer mutierender Stimmungslogiken.

Dieses Kapitel zeigt noch einmal in Kurzform die grundlegenden Programmiermöglichkeiten von statischen und einfachen mutierenden Stimmungslogiken auf.

9.1. Statische Tonsysteme

Ein Tonsystem besteht aus einer Verankerungstaste, einer Menge von Tönen und einem Periodenintervall. Töne stehen in fest definierten intervallischen Zusammenhängen zueinander, wobei mindestens ein Ton eine absolute Frequenz derart zugewiesen bekommt, dass sich mittels der intervallischen Beziehungen für jeden Ton eine eindeutige Frequenz berechnen lässt. Um diese Berechnungen braucht sich der Anwender nicht zu kümmern, er gibt lediglich die Intervallbeziehungen zwischen den Tönen an und gruppiert dann die gewünschten Töne zu Tonsystemen.

9.1.1. Beispiel: Pythagoreïsche Tonleiter auf den weißen Tasten

```
INTERVALL
   Oktave = 2:1
   Quint = 3:2
TON
   c = 260.1
   d = g + Quint - Oktave
   e = a + Quint - Oktave
   f = c - Quint + Oktave
   g = c + Quint
   a = d + Quint
   h = e + Quint
TONSYSTEM
   Pythago = 60 [ c, ,d, ,e,f, ,g, ,a, ,h ] Oktave
```

Die schwarzen Tasten der Klaviatur bekommen keinen Ton zugewiesen, sind also stumm geschaltet[1].

Um dieses Tonsystem im Laufzeitmodul spielen zu können, müssen wir noch eine Stimmungslogik hinzufügen, die unser Tonsystem als Einstimmung benutzt und

[1]Beispielprogramm pythago.mut

ansonsten „leer“ ist. Außerdem definieren wir bei dieser Gelegenheit einen Auslöser, welcher die Logik aktivieren soll, und zwar sinngemäß die Taste P.

```
LOGIK  Pythagoras TASTE P = Pythago [ ]
```

Genaue Definitionen über die Syntax finden Sie im Referenzhandbuch. Hier stehen alle möglichen Sprachkonstruktionen in grafischer Darstellung aufgelistet. Dieses Handbuch legt die Funktionalität der Logiksprache fest. Wenn Sie feststellen, dass eine programmierte Logik nicht so funktioniert, wie Sie es wünschen, so ist das Referenzhandbuch die Entscheidungsinstanz, anhand derer Sie feststellen können, ob das Logikprogramm falsch ist, oder MUTABOR nicht korrekt funktioniert.

Es sei hierbei darauf hingewiesen, dass der MUTABOR-Compiler nicht zwischen Groß- und Kleinschreibung unterscheidet. Die Stimmungslogik

```
LoGiK  PyThAgorAs TAstE p = PytHAgo [ ]
```

ist also mit der obigen völlig identisch.

9.2. Einfache Umstimmungen

Eine Umstimmung definiert eine Veränderung eines oder mehrerer Parameter des aktuellen Tonsystems. Wenn nur ein Parameter verändert werden soll, so benutzen Sie eine einfache Umstimmung, falls Sie mehrere Parameter verändern möchten, so müssen Sie die einzelnen Parameteränderungen als einzelne Umstimmungen programmieren und in einem Umstimmungsbund in der gewünschten Reihenfolge auflisten. Ein Umstimmungsbund darf außerdem Tonsystemnamen beinhalten, wobei bei Ambivalenzen zugunsten der Umstimmung entschieden wird.[2]

Jeder Parameter kann entweder einen neuen, absoluten Wert zugewiesen bekommen, also z. B.

```
UMSTIMMUNG
    Anker49  = 49 [ ]
    Breite17 = [ << 17 >> ]
    TonC     = [ c ]
    PeriodeQ = [ ] Quinte
```

oder der neue Wert soll sich aus dem aktuellen Zustand berechnen:

```
UMSTIMMUNG
    Ankerplus   = @ + 1 [ ]
    Breitehalb  = [ << @/2 >> ]
    Septneu     = [ @,@,@,@,@,@,@,@,@,@,@-Septkomma,@ ]
    Persil      = [ ] @ + 4cent
```

[2] Namensmehrdeutigkeiten werden vom Compiler in Form einer Warnung angezeigt.

Wenn eine Umstimmung die Töne der Fundamentaltonleiter verändern soll, ist natürlich auch eine Mischung von absoluten und relativen Umstimmungen möglich:

```
UMSTIMMUNG
    Toene_neu = [ @, ,@+Komma, ,e,f,fis, @+12Quint-7Oktave]
```

Wenn eine Taste keinen Wert zugewiesen bekommt, so wird sie gesperrt. Es gibt daher zwei Umstimmungsarten, die leicht verwechselt werden:

```
UMSTIMMUNG
    Typ1 = [  ,  ,  ,  ,  ,  ,  ]
    Typ2 = [ @, @, @, @, @, @, @ ]
```

Bei `Typ1` werden die ersten sieben Tasten stumm geschaltet, bei `Typ2` behalten die Tasten ihre Frequenzwerte bei, es ändert sich also nichts.

9.3. Logiken

Im Teil II dieses Handbuches haben Sie bereits einfache Methoden zur Harmonieanalyse[3] kennen gelernt. Zunächst definieren Sie die Harmonien, die bestimmte Aktionen auslösen sollen, z. B.

```
HARMONIE
  C_Dur = { 0, 4, 7 }
  F_Dur = { 5, 9, 0 }
```

In einer Stimmungslogik können Sie nun festlegen, welche Aktion beim Erkennen der entsprechenden Harmonie durchgeführt werden soll, z. B.

```
LOGIK
    Pythagoras_2 Taste P = Pythago [
                            C_Dur -> PythagoC
                            F_Dur -> PythagoF
                           ]
```

[3] Der verwendete Begriff „Harmonieanalyse“ ist hier nicht im musikwissenschaftlichen Sinne gemeint. Vielleicht wäre die Bezeichnung „Klaviaturanalyse“ sinnvoller. Da wir aber in einer Stimmungslogik die „Klaviaturanalyse“ zum Zweck einer – wenn auch sehr eingeschränkten – tatsächlichen Harmonieanalyse eingeführt haben, wollen wir in diesem Handbuch weiterhin den Begriff „Harmonieanalyse“ verwenden, mit dem ausdrücklichen Hinweis, dass damit eigentlich eine „Klaviaturanalyse“ gemeint ist. (Es ist durchaus denkbar, dass in zukünftigen Versionen von Mutabor eine Harmonieanalyse möglich wird, die diesen Begriff auch verdient.)

10. Umstimmungen mit Parametern

Es entstehen häufig Situationen, in denen ein bestimmter Parameter des aktuellen Tonsystems in verschiedenen Umstimmungen verändert werden soll.

Betrachten wir folgenden Ausschnitt aus einem Logikprogramm:

```
LOGIK
    Modulation Taste M = C_Dur [  Taste B -> Anker70
                                  Taste F -> Anker65
                                  Taste C -> Anker60
                                  Taste G -> Anker67
                                  Taste D -> Anker62
                               ]
UMSTIMMUNG
    Anker60 = 60 [ ]
    Anker62 = 62 [ ]
    Anker65 = 65 [ ]
    Anker67 = 67 [ ]
    Anker70 = 70 [ ]
```

Das Tonsystem `C_Dur` ist das bereits aus vielen Beispielen bekannte „reine C-Dur“[1], dieses finden Sie auf Seite 33. Bekanntermaßen ist in diesem Tonsystem z. B. die Quinte d—a keine reine Quinte (3:2). Die Stimmungslogik `Modulation` kann nun per Tastendruck auf einen neuen Bezugston modulieren. Dies geschieht, indem in der entsprechenden Umstimmung die Verankerungstaste neu gesetzt wird, und das heißt, dass die Intervallstruktur des Tonsystems `C_Dur` auf einen neuen Bezugston übertragen wird[2]. Eine solche „Modulation“ verändert also die Töne der Fundamentaltonleiter (bis auf den neuen Grundton), behält aber die Intervallstruktur bei. Im Zustand „C_Dur“ ist die Quinte d—a unrein, wenn Sie aber nach „D_Dur“ oder „G-Dur“ modulieren, so ertönt diese Quinte im Frequenzverhältnis 3:2.

Doch konzentrieren wir uns nun auf die Umstimmungen. Alle fünf Umstimmungen sind *vom selben Typ*: sie weisen der Verankerungstaste einen neuen (absoluten) Wert zu. Da es sehr häufig vorkommt, dass verschiedene Umstimmungen ein und denselben

[1] Wir halten es für wichtig, auch an dieser Stelle zu betonen, dass Stimmungslogiken auf Mutabor in keinster Weise an zwölfstufige Tonsysteme oder gar an Prinzipien der reinen Stimmung gebunden sind. Die Beispiele im Handbuch sind nur der Anschaulichkeit halber meist in Begriffen der zwölfstufigen tonalen Harmonik dargestellt, da wohl die Mehrzahl der Leser mit diesen Strukturen vertraut (wenn auch nicht unbedingt einverstanden) sein dürfte.

[2] Dies lässt sich im Laufzeitmodul sehr schön zeigen, indem Sie sich vor und nach einer Modulation im Protokollfenster das Tonsystem und die Intervallstruktur anzeigen lassen, siehe Begleitbuch “Die Benutzeroberfläche„ .

Parameter verändern, haben wir eine Konstruktion eingeführt, welche alle Umstimmungen gleichen Typs in einer einzigen zusammenfassen kann: *Umstimmungen mit Parametern*[3]. Wir schreiben einfach

```
UMSTIMMUNG
        Anker_neu ( x )  =  x [ ]
```

und aktivieren diese Umstimmung mit verschiedenen Werten für den Platzhalter x:

```
LOGIK
    Modulation Taste M = C_Dur [ Taste B -> Anker_neu(70)
                                 Taste F -> Anker_neu(65)
                                 Taste C -> Anker_neu(60)
                                 Taste G -> Anker_neu(67)
                                 Taste D -> Anker_neu(62)
                               ]
```

Dies gleicht einer einfachen Konstruktion aus der Mathematik: Man definiert eine Funktion, z. B. $f(x) = \sqrt{x+4}$ und erhält daraus später konkrete Ergebnisse, indem man feste Werte einsetzt: $f(12) = 4$. Genauso arbeiten Umstimmungen mit Parametern: man definiert den Typ der Umstimmung und konkretisiert dies *beim Aufruf* in der Logik.

Einen Platzhalter können Sie bei Veränderungen der Verankerungstaste und bei Veränderungen der Breite der Fundamentaltonleiter benutzen. Die vier möglichen Umstimmungen sind also:

```
UMSTIMMUNG
         Anker_absolut(Neu)     = Neu [ ]
         Anker_relativ(Distanz) = @ + Distanz [ ]
         Breite_absolut(Neu)    = [ << Neu >> ]
         Breite_relativ(Faktor) = [ << @ * Faktor >> ]
```

Natürlich sind bei relativen Umstimmungen mit Parametern alle sonst gültigen Rechenzeichen erlaubt (Tatsächlich gibt es also acht verschiedene Typen).

Auf der rechten Seite des Gleichheitszeichens dürfen nur solche Namen benutzt werden, die auch in der Klammer links stehen, da sonst keine eindeutige Zuordnung möglich ist. Die Umstimmung

```
    UMSTIMMUNG Versuch_es_doch (Meier) = @ + Schmitt [ ]
```

führt zu einer Fehlermeldung des Compilers, da der Platzhalter `Schmitt` nirgendwo definiert worden ist.

[3] Im folgenden bezeichnen die Begriffe 'Parameter' und 'Platzhalter' denselben Sachverhalt. An einigen Stellen benutzen wir deshalb den Ausdruck 'Platzhalter', um ihn begrifflich von den 'Parametern' eines Tonsystems zu unterscheiden.

11. Differenziertere Harmonieanalyse

11.1. Harmonieformen

Die Harmonieform ist eine 'translative' Verallgemeinerung der Harmonie auf den gesamten Bereich der Projektionstonleiter. Wenn die Harmonie-*Form* { 0,4,7 } in einem Tonsystem der Breite 12 und der Verankerungstaste 60 (=c') als Auslöser z. B. für eine Umstimmung eingetragen wurde, so wird diese Umstimmung nicht nur beim Erkennen der Harmonie {0,4,7} durchgeführt, sondern ebenfalls beim Erkennen der Harmonie {1,5,8} , sowie {2,6,9}, ... und auch über die Breite der Fundamentaltonleiter hinaus, dann aber modulo derselben: {9,1,4}

Tonal interpretiert stellt die Harmonie {0,4,7} innerhalb eines Tonsystems der Breite 12 und der Verankerungstaste 60 (=c') einen C-Dur-Dreiklang dar, während die Harmonie*form* {0,4,7} als Auslöser auf jeden Dur-Akkord (F-Dur, Cis-Dur, ...) reagiert.

Denken wir uns nun eine Stimmungslogik aus, die (ausgehend von einer gleichstufigen Temperierung) bei einem beliebigen Durakkord alle Töne um ein syntonisches Komma erhöht, und bei einem beliebigen Mollakkord alle Töne um dasselbe erniedrigt. Die Umstimmungen hierzu sind sehr einfach[1]:

```
INTERVALL
        SynKomma = 81:80
UMSTIMMUNG
        Hinauf = [@+SynKomma,@+SynKomma,@+SynKomma,@+SynKomma,
                  @+SynKomma,@+SynKomma,@+SynKomma,@+SynKomma,
                  @+SynKomma,@+SynKomma,@+SynKomma,@+SynKomma]
        Hinab  = [@-SynKomma,@-SynKomma,@-SynKomma,@-SynKomma,
                  @-SynKomma,@-SynKomma,@-SynKomma,@-SynKomma,
                  @-SynKomma,@-SynKomma,@-SynKomma,@-SynKomma]
        Anker60 = 60 [ ]
        Breite12 = [<<12>>]
        Init = { Anker60, Breite12 }
```

Zusätzlich enthält diese Umstimmungsdeklaration die Umstimmungen `Anker60`, `Breite12` und den Umstimmungsbund `Init`, welcher diese beiden der Reihenfolge nach durchführt. Zu Beginn der Stimmungslogiken wird der Umstimmungsbund `Init` als Einstimmung aufgerufen, um alle Werte zu initialisieren. Um die Taste c' als Anker der Fundamentaltonleiter einzustellen, wird die Umstimmung `Anker60` aktiviert. Die

[1]Beispielprogramm aufab.mut

Umstimmung `Breite12` wird benötigt, um die Breite der Fundamentaltonleiter auf zwölf Tasten zu expandieren, da die Harmonieanalyse eines Akkordes {0,4,7} nicht in einem Tonsystem der Breite 1, wie es beim Aufruf des Laufzeitmoduls eingestellt ist, möglich ist (siehe Abschnitt „Handhabung von Grenzfällen“). In der Stimmungslogik benutzen wir die Harmonien `Dur` und `Moll`:

```
HARMONIE
   Dur  = { 0, 4, *7, *10 }
   Moll = { 0, 3, *7 }
```

Ein Durakkord besteht hier aus Grundton und großer Terz, mit eventueller Quinte und großer Septime; ein Mollakkord aus Grundton, kleiner Terz und eventueller Quinte.

Würden wir die Stimmungslogik

```
LOGIK AufAb1 Taste A = Init [ Dur  -> Hinauf
                              Moll -> Hinab ]
```

aktivieren, so würde nur bei einem **C**-Dur-Akkord bzw. **c**-moll-Akkord die Hinauf- bzw. Hinab-Umstimmung durchgeführt, nicht aber bei einem Fis-Dur-Akkord oder einem h-moll-Akkord. Nur durch die Angabe, dass die Harmonie als Harmonieform zu analysieren ist, erreichen wir das Ziel, bei *jedem* Dur- bzw. Mollakkord Umzustimmen:

```
LOGIK
       AufAb2 Taste B = Init [ FORM Dur  -> Hinauf
                               FORM Moll -> Hinab ]
```

In diesem Fall gibt es zwölf äquivalente Logikprogramme, die alle exakt den gleichen Prozess durchführen. Es spielt jetzt nämlich keine Rolle mehr, welchen Durakkord wir in der Harmoniedeklaration explizit angeben. Wir hätten, da es sich nur um eine Harmonieformanalyse handelt, ebenso die Harmonien

```
HARMONIE
       Dur  = { 1,5,*8,*11 }
       Moll = { 4,7,*11 }
```

deklarieren können.

Die Stimmungslogik `Aufab1` hingegen würde dabei nicht mehr auf C-Dur bzw. c-moll reagieren, sondern auf Cis-Dur bzw. e-moll! Nur bei Harmonieformen ist also die Wahl des 'Grundtones' der Harmonie zunächst nicht von Bedeutung. Trotzdem ist es der Verständlichkeit halber meistens besser, einfache Harmonien zu wählen.

11.2. Der ausgezeichnete Parameter „ABSTAND“

In „ABSTAND“ steht unmittelbar nach einer Harmonieformanalyse die Anzahl von Verschiebungsschritten, die notwendig waren, um das Muster der Projektionstonleiter auf die verglichene Harmonie abzubilden. Verdeutlichen wir uns dies an einem Beispiel:

Das aktuelle Tonsystem sei (wieder einmal) das Tonsystem C_Dur mit der Verankerungstaste 60 (=c’) und der Breite 12. Eine Harmonieformanalyse der Harmonie {0,3,7} führt bei jedem beliebigen Mollakkord zu der angegebenen Aktion. Eine Stimmungslogik der Form

```
HARMONIE
      Moll = {0,3,7}
LOGIK
      Xantippe Taste X = C_Dur [
                                   FORM Moll -> Umst ( 7 )
                               ]
```

führt beim Erkennen eines beliebigen Mollakkordes die Umstimmung Umst durch und übergibt an deren ersten Platzhalter den Wert 7. Wir haben hier also einen stets konstanten Übergabewert vorliegen. Mit dem ausgezeichneten Parameter ’ABSTAND’ haben wir einen variablen Übergabewert, welcher an den entsprechenden Platzhalter der Umstimmung genau die Anzahl von Verschiebungen übergibt, die nötig waren, um die Projektionstonleiter auf die abgefragte Harmonie abzubilden. Was bedeutet dies?

Nehmen wir an, die Stimmungslogik Xantippe sei aktiv, und wir schlagen einen c-moll-Akkord an. Um die Projektionstonleiter (in ihr steht die Harmonie {0,3,7}) auf die abgefragte Harmonie Moll abzubilden ist keine Verschiebung nötig, in dem ausgezeichneten Parameter ’ABSTAND’ steht nun also der Wert 0. Dieser Wert wird an die Umstimmung Umst übergeben. Wir wollen an dieser Stelle noch nicht darauf eingehen, was diese Umstimmung macht, sondern nur ermitteln, mit welchem Übergabewert sie aufgerufen wird. Wenn nun ein d-moll-Akkord angeschlagen wird (die Projektionstonleiter hat nun den Inhalt {2,6,9}), so sind *zwei* Verschiebungen nötig, um die PT auf die abgefragte Harmonie (= {0,3,7}) abzubilden, also beide Harmonien ’zur Deckung zu bringen’. Bei einem angeschlagenen fis-moll-Akkord hat ABSTAND den Wert 6, bei b-moll den Wert 10[2].

Was passiert nun, wenn sich die Verankerungstaste des aktuellen Tonsystem verändert, also nicht mehr auf c’ steht? In diesem Fall bezieht sich die Anzahl nötiger Verschiebungen nicht mehr auf die Taste 60, sondern eben auf die neue Verankerungstaste. Setzen wir in unserem Beispiel die Verankerungstaste auf 62 (=d’), z. B. durch eine Umstimmung, die durch einen Tastendruck ausgelöst wurde[3]:

```
HARMONIE
  Moll = {0,3,7}
UMSTIMMUNG
```

[2]Dies heißt natürlich, dass die Verschiebungen zyklisch durchgeführt werden, da der b-moll-Akkord nicht die Harmonie {10,13,17} hat, sondern {10,1,5}. Gerechnet wird also immer modulo 12, bzw. allgemein modulo ’Breite’.

[3]Beispielprogramm xantippe.mut

```
  Anker62 = 62 [ ]
LOGIK
  Xantippe Taste X = C_Dur [ FORM Moll -> Umst(ABSTAND)
                             Taste A   -> Anker62
                           ]
```

Der Anker des aktuellen Tonsystems sei also 62. Welchen Wert erhält ABSTAND bei einem angeschlagenen d-moll-Akkord? Natürlich den Wert 0, da die zu vergleichende Harmonie {0,3,7} mit der 0 immer die Verankerungstaste identifiziert. Ein fis-moll-Akkord setzt ABSTAND auf 4, ein c-moll-Akkord auf 10.

Bei einer Harmonieformanalyse ist der Wert, der in ABSTAND festgehalten wird, ist also immer gleich der Anzahl Verschiebungen bezogen auf die momentane Verankerungstaste. Tonal interpretiert liefert uns ABSTAND die *Stufe*, auf der der angeschlagene Akkord innerhalb der momentanen Tonart (durch den Anker angegeben) steht.

Die Benutzung von ABSTAND ist nur unmittelbar nach einer Harmonieformanalyse sinnvoll, da zu einem späteren Zeitpunkt kein Bezug mehr zur zuletzt erkannten Harmonieform existiert; dennoch kann ABSTAND theoretisch auch ohne Harmonieform-Auslöser benutzt werden, z. B.

```
LOGIK Meier Taste M = [ Taste X -> Umst(ABSTAND) ]
```

Eine sinnvolle Anwendung findet der ABSTAND-Parameter bei der Programmierung einer besonderen mutierenden Stimmung, dem Wandern im tonalen Netz. In der Beschreibung der Demonstrationslogiken im Anhang finden Sie eine Kurzbeschreibung des Prinzip des tonalen Netzes.

Dieser Sachverhalt lässt sich in einer sehr kompakten Form programmieren[4]:

```
LOGIK
       Tonales_Netz Taste T = C_Dur
           [ FORM Dur  -> Transponiere(ABSTAND)
             FORM Moll -> Transponiere(ABSTAND) ]
UMSTIMMUNG
       Transponiere(Distanz) = @ + Distanz [ ]
HARMONIE
       Dur  = {0,4,7}
       Moll = {0,3,7}
```

In diesem Programmierbeispiel ist die Harmonieformanalyse noch nicht sehr flexibel, da sie nur auf vollständig angeschlagene Dur- bzw. moll-Akkorde reagiert. Doch gerade diese Tatsache lässt sich ausnutzen, um Ihnen einen Eindruck davon zu vermitteln, wie die Harmonieanalyse bei MUTABOR vonstatten geht. Wenn Sie nämlich die Logik 'Tonales_Netz' aktivieren und die Quinte c—g anschlagen, so wird diese im Frequenzverhältnis 3:2 (rein) intoniert. Aufgrund der Konstruktion des tonalen

[4]Beispielprogramm mininetz.mut

Netzes ist aber die Quinte d—a nicht rein. Diesen Effekt sollten Sie einmal ausprobieren. Lassen sie die Quinte d—a liegen und schlagen Sie nun ganz kurz den Ton Fis an. In diesem Moment erkennt MUTABOR die Harmonie {2,6,9} als eine um 2 Tasten verschobene Harmonieform von {0,4,7} und stimmt um, indem die Intervallstruktur des reinen C-Dur nun auf den Ton D übertragen wird. Sofort wird die Frequenz des Tones a korrigiert und die Quinte d—a ist rein gestimmt.

Für eine Anwendung in der Praxis ist es jedoch wünschenswert, dass auch Quinten und Terzen erkannt werden. Hierzu müssen wir die Harmonien neu deklarieren[5]:

```
HARMONIE
 Quinte = {0,7}
 Dur    = {0,4,*7}
 Moll   = {0,3,7}
LOGIK
 Netz Taste N = C_Dur
     [ FORM Quinte -> Transponiere(ABSTAND)
       FORM Dur    -> Transponiere(ABSTAND)
       FORM Moll   -> Transponiere(ABSTAND) ]
```

Nun werden Quinten und Terzen sofort erkannt und rein gestimmt.[6] Sie spielen nun in einem mutierenden Tonsystem der reiner Stimmung. Dies ist ein Beispiel dafür, dass mutierende Stimmungen nicht zwangsweise etwas exotisches sind, sondern durchaus auch in Bereichen tonaler Harmonik angewandt werden können.

[5]Beispielprogramm netz.mut

[6]Hierbei ergibt sich ein sehr netter Effekt: man schlage die Terz c—e an. Diese wird von der Harmonieanalyse erkannt und rein gestimmt. Nun nehme man das gis hinzu. Aufgrund der Struktur des tonalen Netzes wird dieses als as interpretiert und viel höher als die reine Terz über e intoniert. Sobald aber das c losgelassen wird, erkennt der Computer wieder das Tastenmuster einer Terz und stimmt diese rein: das as wird zum gis. Nun können wir dasselbe Spielchen wiederholen, indem wir e—gis—c anschlagen (eigentlich e-gis-his) und dann das e loslassen, und so weiter. auf diese Weise sackt die Stimmung immer weiter ab.

12. Anweisungen, Auslöser und Aktionen

Die allgemeinste Form einer Stimmungslogik lautet

```
LOGIK name Auslöser = Einstimmung [ Anweisungen ]
```

Eine Anweisung hat immer die Form

```
Auslöser -> Aktion
```

Bisher haben Sie bereits einige verschiedene Auslösertypen kennen gelernt, nämlich Tastendruck, Harmonie und Harmonieform, sowie den Aktionstyp Aufruf eines Tonsystems/Umstimmung/Logik. In diesem Kapitel werden Sie noch weitere Auslöser und Aktionen, sowie Erweiterungen bereits bekannter Konstruktionen kennen lernen.

12.1. Umstimmungsbünde mit Parametern

übertragen wir nun die Prinzipien der Umstimmung mit Parametern auf Umstimmungsbünde und verdeutlichen dies an einem einfachen Beispiel:

Eine Umstimmung soll die Verankerungstaste auf einen neuen, als Platzhalter übergebenen Wert setzen und anschließend – sozusagen „sicherheitshalber“ – die Breite der Fundamentaltonleiter auf den Wert 9 setzen. Wenn die Breitenkorrektur nicht notwendig wäre, könnten wir einfach

```
UMSTIMMUNG Mephisto(Faust) = Faust [ ]
```

schreiben. Da aber außerdem noch der Parameter 'Breite' verändert werden soll, müssen wir einen Umstimmungsbund benutzen. Ein Umstimmungsbund kann – genau wie eine normale Umstimmung – Platzhalter benutzen. Diese werden hinter dem Namen des Umstimmungsbundes in runden Klammern angegeben. Ein Umstimmungsbund dient nur als Zwischenstation der tatsächlichen Umstimmung, da die eigentlichen Umstimmungen nicht im Bund, sondern in den einzelnen Umstimmungen passieren. Der Umstimmungsbund kann die ihm übergebenen Platzhalter also nicht selber benutzen, sondern nur an die Einzelumstimmungen weitergeben. Wir können nun unseren Umstimmungsbund formulieren:

```
UMSTIMMUNG
        Anker_neu_setzen(wert) = wert [ ]
        Breite_fest_setzen     = [ << 9 >> ]
        Mephisto(Faust)        = { Anker_neu_setzen(Faust),
                                   Breite_fest_setzen }
```

Überlegen wir uns nun, was passiert, wenn wir von einer Logik aus diesen Umstimmungsbund mit dem Übergabewert 6 aufrufen, z. B. durch Drücken der Taste X bei folgender (als aktiv angenommener) Stimmungslogik:

```
LOGIK Goethe Taste G = [ Taste X -> Mephisto(6) ]
```

Die Umstimmung `Mephisto` werde mit dem Übergabewert 6 aufgerufen. Im Platzhalter 'Faust' wird also der Wert 6 festgehalten. Die Angaben im Umstimmungsbund besagen nun, dass zunächst die Umstimmung `Anker_neu_setzen` mit dem in 'Faust' gespeicherten Wert – in unserem Fall 6 – durchgeführt werden soll. Was passiert nun in der Umstimmung `Anker_neu_setzen` ? Diese wird mit dem Übergabewert 6 aufgerufen, welcher dann im Platzhalter 'wert' vermerkt wird; und laut Vorschrift wird die Verankerungstaste auf eben diesen Wert gesetzt. Wenn dies geschehen ist, so ist nun gemäß unseres Umstimmungsbundes die Umstimmung `Breite_fest_setzen` an der Reihe. Diese setzt die Breite der FT absolut auf den Wert 9.

Dieses Prinzip lässt sich nun um noch eine Stufe erweitern. Ein Umstimmungsbund kann auch *mehrere* Platzhalter verwenden. Dies ist notwendig, wenn z. B. nacheinander zwei Umstimmungen aufgerufen sollen, die ihrerseits Platzhalter benutzen. Soll z. B. zunächst die Verankerungstaste auf einen (als Platzhalter zu übergebenden) Wert gesetzt werden, und danach die Breite der Fundamentaltonleiter neu eingestellt werden, und zwar ebenfalls variabel mit Platzhalterübergabe, so muss der Umstimmungsbund, der beide Umstimmungen nacheinander durchführt, zwei Platzhalter benutzen, deren Werte dann an die Umstimmungen weitergegeben werden. Im Programmtext also:

```
UMSTIMMUNG
         Anker_neu ( wert )  = wert [ ]
         Breite_neu ( wert ) = [ << wert >> ]
         Aufruf(para1,para2) = { Anker_neu(para1),
                                 Breite_neu(para2) }
```

Wenn dieser Umstimmungsbund in einer Logik aufgerufen werden soll, so müssen (zwingend!) auch *zwei* Werte übergeben werden, z. B.:

```
LOGIK Meier Taste M = [ Taste X -> Aufruf(64,12) ]
```

Das Drücken der Taste X bewirkt also nacheinander das Neusetzen des Ankers auf 64 und der Breite auf 12.

Ein Umstimmungsbund kann beliebig viele Platzhalter benutzen, sofern sichergestellt ist, dass

- jeder rechts vom Gleichheitszeichen benutzte Platzhalter auch links in den Klammern angegeben wurde,
- der Aufruf mit genau der Zahl an Werten passiert, wie Platzhalter angegeben sind.

Ferner gelten folgende Freiheiten:

- Die Reihenfolge, in der die Platzhalter in den Klammern angegeben wurden, muss nicht mit der Reihenfolge übereinstimmen, in der diese benutzt werden.
- Ein Platzhalter darf mehrmals für Aufrufe benutzt werden.

Alle folgenden Umstimmungsbünde sind syntaktisch korrekt (wenn auch praktisch unbrauchbar):

```
UMSTIMMUNG
    Anker (neu) = neu [ ]
    Breite(neu) = [<< neu >>]
    Periode     = [ ] Quint

    Simpel(x,y)={ Breite(y) , Anker (x) }
    Naja(a,b,c)={ Breite(b) , Anker (c), Periode, Anker(b),
                  Breite(a) }
```

12.2. Auswählende Umstimmungsbünde

Ein Umstimmungsbund ist eine sequenzielle Auflistung von Umstimmungen[1], die nacheinander durchgeführt werden sollen. Wir wollen nun noch eine weitere Konstruktion eines Umstimmungsbundes einführen: den *auswählenden Umstimmungsbund*.

Ein auswählender Umstimmungsbund ist ein Umstimmungsbund, dem ein besonderer Parameter übergeben wird, der ein Auswahlkriterium dafür darstellt, *welche* der im Bund aufgelisteten Umstimmungen durchgeführt werden sollen. Dieser Parameter ist (wie gewöhnlich) eine ganze Zahl, die durch die aufrufende Logik entweder explizit angegeben oder in der Form des variablen ABSTAND-Parameter übergeben wird.

Innerhalb des auswählenden Umstimmungsbundes stehen dann Wenn-Dann-Zuweisungen, die angeben, welche Umstimmungen bei welchem Wert des Auswahlparameters durchgeführt werden sollen. Doch nun erst einmal ein Beispiel:

```
UMSTIMMUNG
    Auswahl ( X ) = X { 1 -> Susi
                        2 -> Otto
                        3 -> Fritz
                        7 -> Irene }
```

Wenn dieser Umstimmungsbund mit dem Übergabewert 3 aufgerufen wird, also `Auswahl(3)` , so wird die Umstimmung `Fritz` durchgeführt. Bei einem Übergabewert

[1] oder Tonsystemen

von 7 wir entsprechend Irene aktiviert. Jeder Wert, der nicht 1,2,3 oder 7 ist bewirkt keine Umstimmung.

Die Syntax dieser Konstruktion erinnert an eine Mischung aus Stimmungslogik und Umstimmungsbund. Und genau das stellt der auswählende Umstimmungsbund auch dar. Wieder haben wir eine Art „Auslöser" und eine „Aktion", nämlich den Vergleichswert und die angegebene Umstimmung. Vor der geschweiften Klammer wird der Name des Parameters angegeben, der mit den in den Klammern angegebenen Werten verglichen werden soll.

Diese Konstruktion können wir nun erweitern, denn es kann durchaus erwünscht sein, dass auch der auswählende Umstimmungsbund mehr als nur einen Parameter bekommt, und alle bis auf den auswählenden Parameter an die aufgelisteten Umstimmungen weiter gibt. Hier wird ganz analog den mehrparametrigen Umstimmungsbünden verfahren, und dabei bekommt die explizite Angabe des Namens des Auswahlparameters vor den geschweiften Klammern ihren Sinn:

```
UMSTIMMUNG
   Transpo ( X , Wohin ) = X { 0 -> Transponiere_hoch(Wohin)
                               1 -> Transponiere_runter(Wohin) }
```

Wird diese Umstimmung mit den Übergabewerten Transpo(1,6) aufgerufen, so wird die Umstimmung Transponiere_runter mit dem weitergereichten Übergabewert 6 durchgeführt.

Wie in einem Umstimmungsbund können Sie auch mehrere Umstimmungen auflisten, die beim Erkennen eines Vergleichswertes nacheinander ausgeführt werden sollen. Dies erspart die Programmierung eines zusätzlichen Umstimmungsbundes, falls mehrere Parameter verändert werden sollen:

```
UMSTIMMUNG
    Auswahl(X) = X { 0 -> Halb1,Expandiere
                     1 -> Reset
                     2 -> Reset
                     3 -> Reset }
```

Der auswählende Parameter muss nicht unbedingt als erster in der Parameterliste angegeben werden. Deshalb wird sein Name vor die geschweifte Klammer geschrieben. Außerdem können Sie in einem auswählenden Umstimmungsbund auch die ANSONSTEN–Option benutzen. Dies bedeutet, dass die der ANSONSTEN–Option zugeordneten Umstimmungen durchgeführt werden, wenn kein anderer Vergleich passt:

```
UMSTIMMUNG
    Verschiebe(X) = X { 0         -> Halb1,Expandiere
                        ANSONSTEN -> Transponiere }
```

Hier würden also im Falle von X=0 nacheinander die Umstimmungen `Halb1` und `Expandiere` durchgeführt, und *ansonsten* immer die Umstimmung `Transponiere`.

Insbesondere im Zusammenhang mit dem ABSTAND–Parameter und der ANSONSTEN-Option bildet der auswählende Umstimmungsbund vielfältige Möglichkeiten. Man denke z. B. an eine mutierende Stimmung, die Septime des Dominantseptakkordes zu groß nehmen soll. Mit Hilfe des auswählenden Umstimmungsbundes ist dies kein Problem[2]:

```
LOGIK
     Septbetonung Taste S = Halb12 [ FORM Sept -> Betone(ABSTAND) ]

HARMONIE
     Sept = { 0,4,*7,10 }

UMSTIMMUNG
     Betone(dist) = dist { 7         -> GrosseSept
                           ANSONSTEN -> Halb12 }

     GrosseSept = [ @,@,@,@,@,@+Faktor ]

INTERVALL
     Faktor = 24 Wurzel 2

UMSTIMMUNG
     Halb12     = { Halb,Expandiere }
     Expandiere = [ << 12 >> ]

TON
     a = 440

TONSYSTEM
     Halb = 69 [ a ] 2 Faktor
```

12.3. Kommunikation – MIDIOUT und MIDIIN

Normalerweise verarbeitet Mutabor nur die MIDI-Informationen 'NOTE ON' und 'NOTE OFF', also nur Informationen darüber, welche Tasten auf der Klaviatur angeschlagen bzw. losgelassen wurden. Alle anderen MIDI-Informationen, z. B. Program Change – Meldungen, etc. . . werden ignoriert.

Sie können jedoch solche Meldungen gezielt als Auslöser verwenden. Auf diese Weise können Sie Mutabor von Ihrem Masterkeyboard aus „fernsteuern“. Oder

[2]Beispielprogramm auswahl.mut

Sie können mit Hilfe eines Sequenzer-Programms auf einem zweiten Computer alle Umstimmvorgänge gezielt beeinflussen.

Der Auslöser MIDIIN kann an allen Stellen eingesetzt werden, wo ein Auslöser erforderlich ist, also als Auslöser einer Stimmungslogik oder als Bestandteil einer Anweisung innerhalb einer Stimmungslogik. Ein Beispiel:

```
LOGIK Otto MIDIIN ( #C0, #06 ) = C_Dur [ ]
```

Das Doppelkreuz # bedeutet, dass die darauf folgende Zahl zur Basis 16 interpretiert werden soll (Sedezimal, Hexadezimal). #C0 entspricht also (dezimal) 182.[3] In diesem Beispiel ist der Auslöser für die Logik Otto der MIDI-Event #C0 #06. Das bedeutet, dass die Logik Otto aktiviert wird, wenn über MIDI die Zahlenfolge 182, 6 an MUTABOR geschickt wurde. Auf diese Weise können Sie z. B. Program Change-Meldungen benutzen, um Logiken umzuschalten oder Umstimmungen durchzuführen. Die meisten Masterkeyboards können gezielte Program Change Meldungen senden, so dass diese richtiggehend ideal zum Umschalten von Logiken und Umstimmungen benutzt werden können.

Das Gegenstück zum MIDIIN-Auslöser ist die MIDIOUT-Aktion. Hier werden die angegebenen Daten direkt an die MIDI-Schnittstelle ausgegeben. Damit können dem angeschlossenen Synthesizer/Sampler Spezialkommandos z. B. zur Änderung der Klangfarbe übermittelt werden. Die Syntax ist analog zur MIDIIN-Anweisung:

```
LOGIK Meier Taste M = [ Taste Q -> MIDIOUT(160,#54,23) ]
```

Hier wird, wenn die Logik Meier (durch Drücken der Taste M) aktiv ist, beim jedem Drücken der Taste Q an die MIDI-Schnittstelle die Zahlenfolge 160, 84[4] und 23 ausgegeben.

Bei dem Auslöser `MIDIIN` unterliegt die Folge von MIDI-Bytes folgenden Einschränkungen im Sinne des MIDI-Standards:

- Die erste Zahl muss ein Statusbyte sein, also im Bereich zwischen 128 und 255 liegen. Bei MIDIIN ist es außerdem nur sinnvoll, Zahlen aus den Bereich zwischen 160 und 239 zu wählen, da die MIDI-Bytes 128 bis 159 für NOTE-ON bzw. NOTE-OFF-Meldungen reserviert ist und nicht in die Analyse miteinbezogen wird. Außerdem sind Systemmeldungen (größer 239) ebenfalls von der Analyse ausgeschlossen.

- Das untere Nibble (die unteren 4 Bits) des Statusbytes muss 0 sein; falls es das nicht ist, wird der Wert vom Compiler entsprechen korrigiert und eine Warnung ausgegeben. Praktisch gelten also nur die Zahlen #A0, #B0, #C0,

[3] In der MIDI-Dokumentation sind die Angaben meistens im hexadezimalen Zahlenformat angegeben, und um Ihnen das Umrechnen zu ersparen, können Sie MIDI-Bytes direkt im Hex-Format eingeben.

[4] Entspricht hexadezimal #54 !

#D0 und #E0[5]. Dies ist für die MIDI-Kanal-Zuordnung wichtig (siehe weiter unten, Abschnitt „Simulation verschiedener Instrumente“).

- Alle nachfolgenden Zahlen müssen Datenbytes, also kleiner als 128 sein. Negative Zahlen sind als MIDI-Byte natürlich nicht zulässig.

Die Byte-Folgen bei der Aktion `MIDIOUT` unterliegen nur der Einschränkung, dass sie im Bereich von 0 – 255 liegen müssen.

Denkbar ist eine *Zusammenschließung (Vernetzung) mehrerer Computer*, von denen einer als „Master“ alle Vorgänge mit einem Sequenzer-Programm kontrolliert. Die übrigen Rechner sind mit je einem Synthesizer/Sampler verbunden. Mittels der MIDI-Kommunikation durch den MIDIIN-Auslöser und die MIDIOUT-Aktion können sich die auf diese Weise vernetzten MUTABOR-Systeme aufeinander einstimmen und Informationen austauschen. So können Sie pro Rechner eine 16-fache mikrotonale Polyphonie erzeugen – bei nur drei Computern (und Samplern) stehen Ihnen damit schon fast unbegrenzte Möglichkeiten der Klangfülle zur Verfügung, nämlich 48 Stimmen!

Um gezielte Nachrichten über gerade geschehene Umstimmungen an weitere MUTABOR-Geräte in der Kette senden zu können, ist es erlaubt, MIDIOUT-Nachrichten auch in Stimmungsbünde einzubauen:

```
UMSTIMMUNG
  Hoch_hinaus = [ @+Terz, @+Terz, @+Terz, @+Terz, @+Terz ]
  Aber_schmal = [ ] @ - 10 cent

  Umstimmungsbund_mit_Nachricht = { Hoch_hinaus ,
                                    MIDIOUT ( #B0, #05, #60 ) ,
                                    Aber_schmal }
```

Wenn nun die Umstimmung `Umstimmungsbund_mit_Nachricht` aufgerufen wird, so werden zunächst die ersten 5 Töne der Fundamentaltonleiter um das Intervall Terz (was auch immer das sein mag) erhöht, dann die MIDI-Nachricht #B0,#05,#60 (=Controller 5 auf #60 setzen) geschickt und anschließend das Periodenintervall um 10 Cent verkleinert.

[5] Diese anscheinende Beschränkung auf nur fünf verschiedene MIDI-Auslöser trügt. Da diesen Statusbytes nämlich noch ein bis zwei Datenbytes folgen, beträgt die Anzahl verschiedener analysierbarer MIDI-Kommandos weit über 60000, was wohl völlig ausreichend ist.

13. Noch mehr über Aufrufe

13.1. Direkte Umstimmungsbünde

Manchmal erscheint es zu umständlich, einen Umstimmungsbund zu deklarieren, falls ein Auslöser mehrere Aktionen auslösen soll. Wir haben deshalb in die Logiksprache ab der Version 2.1 die Möglichkeit eingebaut, Umstimmungsbünde direkt hinter den Auslöserpfeil in einer Logik zu schreiben. Die beiden folgenden Logiken `Entweder` und `Oder` arbeiten völlig identisch und unterscheiden sich nur in der syntaktischen Formulierung:

```
INTERVALL
  Viertelton = 24 wurzel 2

TON
  a' = 440

TONSYSTEM
  Viertel = 69 [ a' ] Viertelton

UMSTIMMUNG
  Allerhand = { MIDIOUT(254),
                Viertel,
                MIDIOUT(#C0,#2B),
                Expandiere }
HARMONIE
  Grunz = { 1,*2,3 }

LOGIK
  Entweder Taste A = [ FORM Grunz -> Allerhand ]
  Oder     Taste B = [ FORM Grunz ->  { MIDIOUT(254),
                                        Viertel,
                                        MIDIOUT(#C0,#2B),
                                        Expandiere } ]
```

Die Konstruktion eines „direkten Umstimmungsbundes“ arbeitet die innerhalb der geschweiften Klammern aufgeführten Aktionen – genau wie ein normaler Umstimmungsbund – sequentiell ab.

Da in komplexen Stimmungslogiken meistens mehrere Klammerebenen ineinander geschachtelt werden, sollten Sie sehr sorgfältig prüfen, dass jede geöffnete Klammer

auch an der richtigen Stelle die korrespondierende geschlossene Klammer hat.

13.2. Wer darf wen aufrufen ?

Dies ist eine wichtige Frage, auf die es eine einfache Antwort gibt: sofern es keine Selbstbezüglichkeiten gibt, darf quasi jeder jeden aufrufen, also:

- (einzige Ausnahme:) Die Einstimmung einer Logik darf nur ein Tonsystem oder eine Umstimmung (inkl. Umstimmungsbund) aufrufen.
- Ein Umstimmungsbund darf eine Umstimmung, ein Tonsystem, eine MIDIOUT-Nachricht und sogar eine Logik aufrufen.
- Ein direkter Umstimmungsbund darf natürlich alles, was ein Umstimmungsbund darf.
- Eine Aktion darf ebenfalls eine Umstimmung, ein Tonsystem, eine MIDIOUT-Nachricht und eine Logik aufrufen.

Dabei können theoretisch gewisse pathologische Fälle auftreten, z. B. die Logik, die in ihrer Einstimmung einen Umstimmungsbund aufruft, welcher eine MIDI-Nachricht sendet und anschließend – denn dem Umstimmungsbund ist dies ja gestattet – eine Logik aktiviert, die ihrerseits in der Einstimmung. . . also wie gesagt: pathologisch. Da solche Fälle jedoch keinerlei musikalische Relevanz haben, wird hier nicht weiter darauf eingegangen, dem Benutzer wird außerdem geraten, solche Fälle nicht auszuprobieren. . .

14. Beschreibung interner Vorgänge

Um komplexe Stimmungslogiken gezielt einsetzen zu können, kann es sehr wichtig werden, genau zu wissen, wie das Laufzeitmodul intern arbeitet. Was passiert z. B., wenn eine Änderung der Breite auf 0 verlangt wird, oder wie werden die Frequenzen liegender Töne korrigiert? Mit diesen Fragen befasst sich das nun folgende Kapitel.

14.1. Grundsätzliches

Wenn Sie eine Stimmungslogik programmiert haben und diese im freien Spiel anwenden, so können Sie sich jederzeit den momentanen Status des Instruments anzeigen lassen. Diese Funktionen sind im Beiheft „Die Oberfläche“ beschrieben.

14.2. Handhabung von Grenzfällen

Es kann zur Laufzeit passieren, dass gewisse Parameter unzulässige Werte einnehmen sollen. „Was passiert, wenn…“ lautet die Frage; es ist sicher, dass sich das aktuelle Tonsystem nach dem hundertsten Aufruf der Umstimmung

```
UMSTIMMUNG
                Aufi_gehts = [ @ + Halbton ]
```

oder ähnlichen, meist harmloseren Konstruktionen nicht mehr wie vorgeschrieben verhalten kann.

Die Breite der Fundamentaltonleiter kann nicht kleiner als eins und nicht größer als 60 werden. Wenn eine Umstimmung diese Grenze unter- bzw. überschreiten möchte, so wird der Umstimmungsauftrag schlichtweg ignoriert. Ein Tonsystem der Breite null ist wohl auch theoretisch undenkbar, und die Beschränkung auf eine maximale Breite von 60 Tasten – das sind immerhin fünf Oktaven – dürfte in der Praxis keine Schwierigkeiten bereiten und ist eine rein technische Grenze.

Tonhöhen. Der tiefste Ton, den Mutabor ausgeben kann, entspricht der MIDI-Taste 0 (das sind etwa 19 Hz), der höchste Ton entspricht der MIDI-Taste 127 (etwa 12000 Hz). Größere oder kleinere Frequenzwerte werden ausgegeben, aber aus technischen Gründen in den hörbaren Bereich projiziert, so dass eine 'MIDI-Frequenz' von 137 mit (137 modulo 128) , also 9 erklingt. Ebenso mit Noten, die tiefer als die 'MIDI-Frequenz' 0 kommen. Mutierende Stimmungen, die über den höchsten Ton hinaus wollen, werden also gelassen und nicht – wie bei der Änderung der Breite – einfach

beschnitten. Dieses geschlossene Verhalten führt manchmal zu lustigen Effekten, insbesondere wenn das Kreisen in diesem Tonraum mittels wenigen Umstimmungen passiert. Wir möchten Ihnen deshalb folgende Stimmungslogik nicht vorenthalten, die ein geradezu chaotisches – aber sehr interessantes – Verhalten an den Tag legt[1]:

```
INTERVALL
 i1=1:1   i2=2:1   i3=3:1   i4=4:1   i5=5:1   i6=6:1
 i7=7:1   i8=8:1   i9=9:1   i10=10:1 i11=11:1 i12=12:1
 i13=13:1 i14=14:1 i15=15:1 i16=16:1 i17=17:1 i18=18:1
 i19=19:1 i20=20:1 i21=21:1 i22=22:1 i23=23:1 i24=24:1
 i25=25:1 i27=27:1 i28=28:1 i30=30:1

TON
 a=110
 o2=a+i2   o3 =a+i3  o4 =a+i4  o5 =a+i5  o6 =a+i6  o7 =a+i7
 o8=a+i8   o9 =a+i9  o10=a+i10 o11=a+i11 o12=a+i12 o13=a+i13
 o14=a+i14 o15=a+i15 o16=a+i16 o17=a+i17 o18=a+i18 o19=a+i19
 o20=a+i20 o21=a+i21 o22=a+i22 o23=a+i23 o24=a+i24 o25=a+i25
 o27=a+i27 o28=a+i28 o30=a+i30
 b=a+5i2
 u2=b-i2   u3=b-i3   u4=b-i4   u5=b-i5   u6=b-i6   u7=b-i7
 u8=b-i8   u9=b-i9   u10=b-i10 u11=b-i11 u12=b-i12 u13=b-i13
 u14=b-i14 u15=b-i15 u16=b-i16 u17=b-i17 u18=b-i18 u19=b-i19
 u20=b-i20 u21=b-i21 u22=b-i22 u23=b-i23 u24=b-i24

TONSYSTEM
 Mischung = 60 [ a,u15,o2,u14,o3,o4,u13,o5,u12,o6,u11,o7,
               o8,u10,o9,u9,o10,o11,u8,o12,u7,o13,u6,o14,
               o15,u5,o16,u4,o17,o18,u3,o19,u2,o20,b,o21] i1

HARMONIE
 Alles={0,*1,*2,*3,*4,*5,*6,*7,*8,*9,*10,*11,*12,*13,*14,
    *15,*16,*17,*18,*19,*20,*21,*22,*23,*24,*25,*26,
    *27,*28,*29,*30,*31,*32,*33,*34,*35,*36,*37,*38,
    *39,*40,*41,*42,*43,*44,*45,*46,*47,*48}

UMSTIMMUNG
 Verschiebe(Dist) = @+Dist [ ]

LOGIK
    Chaos Taste C = Mischung
         [ FORM 0~Alles -> Verschiebe(ABSTAND) ]
```

[1]Beispielprogramm chaos.mut

Diese Stimmungslogik berechnet eine Oberton- und eine Untertonreihe ab der tiefsten gedrückten Taste. Sobald sich also die tiefste gedrückte Taste ändert, werden nahezu alle liegenden Tasten umgestimmt; die Stimmung steigt sofort stark in die Höhe, sehr bald wird wieder in einen sehr tiefen Tonbereich gesprungen, ... Die Stimmung ist nur so lange stabil, wie Sie die tiefste Taste konstant lassen. Zum Programmtext dieser Stimmungslogik sollte noch gesagt werden, dass der tabellarisch anmutende Anblick nur durch den Umstand nötig wurde, dass hier tatsächlich *48 verschiedene Töne* deklariert wurden, die auf 24 verschiedenen Intervallen basieren. Ein solches Logikprogramm wirkt zwar wenig elegant, lässt sich aber leider nicht kürzer fassen. Glücklicherweise befindet es sich auf der Beispieldiskette, so dass Sie es nicht abtippen müssen, um in der Genuss dieser exotischen Stimmung zu kommen.

Wenn die Verankerungstaste im Zuge einer relativen Umstimmung größer als 96 werden soll, so wird vom geforderten Wert so oft die Breite der Fundamentaltonleiter subtrahiert, bis die Verankerungstaste kleiner als 96 geworden ist. Falls der Wert kleiner als 36 werden soll, wird entsprechend addiert. Diese Korrektur hat keinerlei hörbaren Effekt und ist nur aus Gründen der internen Rundungsoptimierung notwendig. Ein Tonsystem z. B. der Breite 21 reagiert auf die beiden folgenden Umstimmungen:

```
UMSTIMMUNG
      Auf = @ + 17 [ ]
      Ab  = @ - 4  [ ]
```

ohne hörbaren Unterschied, da in beiden Fällen die gleiche 'Tonigkeit' angesprungen wurde.

Die Anzahl gleichzeitig liegender Töne wird durch das Laufzeitmodul nicht begrenzt. Alle liegenden Tasten, und es spielt dabei keine Rolle, wie viele es derer sind, werden in die Klaviaturanalyse einbezogen. So gesehen würde ein Elefant, der seinen Fuß auf die Klaviatur stellt, eine gültige Harmonie spielen, die auch theoretisch auswertbar ist. Tatsächlich ist aber die maximale Zahl *hörbarer* Töne je nach Ausgabegerät und MIDI-Kanal-Zuordnung[2] begrenzt, gewöhnlicherweise auf sechzehn, manchmal auch auf acht oder vier Töne.

Die Anzahl verschiedener Stimmungslogiken, Tonsysteme und Umstimmungen ist nur durch die Speicherkapazität Ihres Computers beziehungsweise durch die maximale Anzahl verschiedener Auslöser begrenzt. Wenn Sie also keine MIDI- oder Harmonieauslöser benutzen, sondern alle Logiken per Tastendruck aktivieren, so ist deren Anzahl auf die sechsundzwanzig Buchstaben (A bis Z) begrenzt.

[2]Siehe Abschnitt „Simulation verschiedener Instrumente“

Die Anzahl der Ton-Umstimmungen kann möglicherweise nicht mit der Breite des Tonsystems übereinstimmen. Überzählige Ton-Umstimmungen werden dann ignoriert, ebenso bleiben überzählige Töne der Fundamentaltonleiter einfach unverändert.

Harmonieanalyse. Es kann vorkommen, dass eine Harmonieanalyse auf eine Harmonie testen soll, dessen eine Tonigkeit jenseits der Breite der momentanen Fundamentaltonleiter liegt. Was passiert, wenn z. B. die Harmonie `Dur = { 0,4,7 }` in einem Tonsystem der Breite 5 getestet werden soll? Es gibt hier prinzipiell zwei Möglichkeiten: entweder kann eine solche Harmonie im genannten Tonsystem niemals erkannt werden, da es dort keine 7. Tonigkeit gibt, oder alle über die Breite hinausgehenden Tonigkeiten der Harmonie werden ignoriert, in unserem Beispiel also würde praktisch auf die Harmonie `{ 0,4 }` getestet werden. Wir haben uns für die zweite Lösung entschieden, aber erst der Einsatz in der Praxis kann darüber entscheiden, ob nicht vielleicht doch die erste Möglichkeit sinnvoller ist.

14.3. Über Rundungsfehler

Das Laufzeitmodul von Mutabor arbeitet bei der internen Speicherung von Tonfrequenzen mit einer Genauigkeit von 10^{-8} Cent. Es ist zwar möglich, dass bei ständig durchgeführten relativen Umstimmungen unter bestimmten Umständen irgendwann einmal hörbare Fehl-Stimmungen auftreten, die durch Rundungsfehler verursacht wurden, jedoch wird dieser Fall in der Praxis wohl kaum eintreten.

Falls Sie das Gefühl haben sollten, dass das Laufzeitmodul trotz korrekter Stimmungslogik falsch intoniert, so ist es ratsam, erst einmal den internen Zustand mittels der Protokollfunktionen abzufragen und erst bei Nichtübereinstimmung mit der Erwartung und nach eingehender Prüfung des Logikprogramms auf einen Programmfehler in der Mutabor-Software zu schließen. . .

Noch eine Bemerkung über absichtliche Rundungen: wenn ein Tonsystem der Breite 15 der Umstimmung

```
UMSTIMMUNG  Halbieren = [ << @/2 >> ]
```

ausgesetzt ist, so wird der bei der Rechnung entstehende Nachkommawert einfach abgeschnitten. Das resultierende Tonsystem wäre also hier von der Breite 7.

14.4. Arbeitsweise der Synthesizer-Treiber

Leider haben es die Hersteller handelsüblicher Synthesizer versäumt, in das MIDI-Protokoll Befehle zur direkten Ansteuerung von Mikrotönen einzubauen.[3] Wir müs-

[3] Einzige Ausnahme bildet hier der Synthesizer FB-01 von Yamaha. Hier gibt es eine System-Exklusiv- Meldung der Gestalt „Schalte die Note c mit der Feinstimmung +42 Feinstimmeinheiten an“, wobei eine Feinstimmeinheit einem Hundertachtundzwanzigstel Halbton entspricht.

sen daher einen kleinen Trick anwenden, um dem Synthesizer/Sampler trotzdem Mikrotöne entlocken zu können.

14.4.1. Die Entfesselung der Mikrotöne

Die meisten Synthesizer haben einen „Pitch-Bender". Das ist ein Drehrad, mit dessen Hilfe man die Frequenz der Töne während des Spiels anheben oder absenken kann. Und zwar je nach Einstellung am Synthesizer zwischen einem und zwölf Halbtönen. Diese Anhebung erfolgt in Schritten von einem Vierundsechzigstel des Maximalintervalls. Wenn man also die Weite des Pitch-Bendings auf einen Halbton einstellt, so kann man damit eine Auflösung von 1/64 Halbton erreichen, was theoretisch 1,6 Cent entspricht.

Für ein solches Pitch-Bending existiert glücklicherweise auch ein MIDI-Code, mit dem man diese Funktion über MIDI ansteuern kann. Leider verändert eine Pitch-Bend-Nachricht nicht nur einen gezielten Ton, sondern alle momentan liegenden Töne. Auf diese Weise ist es also noch nicht möglich, das c' um 4 Cent zu erhöhen und gleichzeitig das f" um 35 Cent zu erniedrigen.

Doch zu guter Letzt ist auch dieses Problem in den Griff zu bekommen. Wenn man sechzehn Synthesizer hintereinander schaltet, so dass jeder Synthesizer nur *einen* Ton spielt, so könnte man jedem Gerät eine *eigene* Pitch-Bend-Information geben – und das Problem wäre gelöst. Nun hat zwar nicht jeder sechzehn gleiche Synthesizer, aber in diesem Punkt kommt uns die moderne Technik zur Hilfe: die meisten Synthesizer lassen sich in einer „Multi-Mode"-Betriebsart ansteuern. D. h. der Synthesizer tut so, als ob er aus sechzehn eigenständigen Synthesizern gleichen Typs bestehen würde. Man kann nun jeden der sechzehn 'Synthesizer im Synthesizer' gesondert ansprechen, und somit erreichen, dass bis zu sechzehn gleichzeitig liegende Töne jeweils eine *eigene* Feinstimminformation bekommen können.

14.4.2. Unbedingt beachten...

Dabei sind jedoch folgende Dinge von großer Wichtigkeit:

- Auf jedem MIDI-Kanal muss dasselbe Instrument (Klangfarbe) eingestellt werden, damit mehrere gleichzeitig liegende Töne auch mit derselben Klangfarbe erklingen.

- Auf jedem Kanal muss die Reichweite des Pitch-Benders auf 1 Halbton eingestellt werden (bzw. den Wert, den Sie im Setup-Dialog von MUTABOR eingestellt haben), damit die Feinstimmungen korrekt durchgeführt werden. Diese „Pitch Bend Range" ist bei den meisten Synthesizern ein Parameter der Klangfarbe und nicht des Multis.

Leider lässt die Klangqualität dieses Gerätes nach heutigen Maßstäben zu wünschen übrig und außerdem wird es nicht mehr produziert. Wahrlich schade, dass Yamaha diese für unsere Zwecke so sinnvolle Option aus ihrem Programm gestrichen hat. Im Gegensatz zu früheren Versionen von **MUTABOR** wird dieser Synthesizer nicht mehr speziell unterstützt.

Diese Einstellungen erfordern gewisse Kenntnisse über die Bedienung des benutzten Synthesizers, über die dieses Handbuch aufgrund der Mannigfaltigkeit des Angebotes von Synthesizern und Samplern leider keine genaue Beschreibung liefern kann. Glücklicherweise sind nur die beiden oben genannten Einstellungen nötig, und wenn Sie erst einmal wissen, wie Sie Ihren Synthesizer richtig einzustellen haben, dürfte dies kein Problem mehr darstellen. Falls Sie dennoch mit der Einstellung Ihres Synthesizers Schwierigkeiten haben, so stehen wir Ihnen gerne mit Rat und Tat beiseite, denn an solchen technischen Kleinigkeiten sollte der Einsatz von MUTABOR nicht scheitern.

14.4.3. Korrektur von Frequenzen

Eine wichtige Frage ist noch ungeklärt. Wie soll sich MUTABOR verhalten, wenn sich aufgrund einer Umstimmung die Frequenz einer liegenden Note ändern muss. Es steht natürlich außer Frage, dass der Frequenzwert korrigiert werden muss. Und hierzu gibt es zwei verschiedene Fälle:

a) Die korrigierte Frequenz liegt im Einzugsbereich derselben MIDI-Taste (z. B. soll von Taste 54 mit Feinstimmung $\frac{4}{64}$ auf Taste 54 mit Feinstimmung $\frac{9}{64}$ umgestimmt werden). Es muss lediglich die Pitch-Bend-Information mit dem neuen Wert an den Synthesizer geschickt werden. Der (liegen bleibende) Ton wird einfach in seiner Tonhöhe korrigiert.

b) Wenn die korrigierte Frequenz dagegen nicht im Einzugsbereich derselben MIDI-Taste liegt (z. B. von Taste 54 mit $\frac{4}{64}$ auf Taste 53 mit $\frac{62}{64}$), so muss der Ton ausgeschaltet werden und mit der korrigierten Frequenz neu angeschlagen werden.

So klingen die meisten kleinen Frequenzänderungen sehr natürlich, während größere Änderungen durch ein Neuanschlagen des Tones je nach eingestellter Klangfarbe mehr oder minder unangenehm auffallen. Dies ist ein technisches Problem bei der Ansteuerung von Synthesizern durch MIDI, aber kein prinzipielles Problem von MUTABOR.

15. Simulation verschiedener Instrumente

Die meisten modernen Synthesizer oder Sampler können in einem „Multi-Mode" betrieben werden. Hierbei kann jedem MIDI-Kanal eine eigene Klangfarbe zugeordnet werden. Auf diese Weise kann ein einziges Gerät bis zu sechzehn verschiedene Instrumente (Klangfarben) simulieren. Wie Sie im vorigen Kapitel gelesen haben, nutzt Mutabor diese Option aus, um auf diese Weise mikrotonale Polyphonie zu ermöglichen.

Leider beansprucht Mutabor zur Ansteuerung der Mikrotöne für jeden gleichzeitig liegenden Ton einen eigenen MIDI-Kanal[1]. (Ein vierstimmiger Akkord belegt also immer vier MIDI-Kanäle.) Deshalb muss der Synthesizer zum Musizieren mit Mutabor so eingestellt werden, dass er auf jedem MIDI-Kanal dieselbe Klangfarbe spielt[2]. Um diese Reduktion auf nur eine Klangfarbe im Betrieb mit Mutabor zu verhindern, können Sie die Kanalzuordnung selbst definieren. Stellen Sie die Klangfarbenzuordnung an Ihrem Synthesizer/Sampler z. B. so ein, daß die MIDI-Kanäle eins bis acht ein Klavier und die Kanäle neun bis sechzehn ein Streichorchester spielen. Wenn Sie im normalen Betrieb nun einen 12-stimmigen Akkord anschlagen, so werden die ersten acht Töne vom Klavier und die letzten vier vom Streichorchester gespielt. Dies macht offensichtlich keinen Sinn. Nun können Sie aber dem Mutabor-Compiler mitteilen, dass die Kanäle 1–8 vom ersten Instrument belegt werden und die Kanäle 9–16 vom zweiten. Dies reduziert die maximale Polyphonie auf acht Stimmen pro Instrument, ermöglicht aber dafür die Simulation mehrerer Instrumente.

Sie können so z. B. ein Streichquartett oder eine beliebige andere Kombination verschiedener Instrumente simulieren.

Sie treffen in der MidiKanal-Deklaration eine Zuordnung, welcher MIDI-Eingabekanal welchen MIDI-Ausgabekanälen zugeordnet werden soll. Wenn Sie keine MidiKanal-Deklaration angeben, so wird diese Zuordnung mit

```
MIDIKANAL
   1 -> 1-16
```

initialisiert. Das bedeutet, dass alle NOTE-ON bzw. NOTE-OFF-Meldungen, die über die MIDI-Schnittstelle auf MIDI-Kanal 1 vom Masterkeyboard empfangen werden, an Kanäle 1 bis 16 weitergeleitet werden.

[1] Ausnahme: der FB-01-Synthesizer von Yamaha, siehe Fußnote auf Seite 90

[2] Wenn Sie den vorigen Abschnitt noch nicht gelesen haben, so sollten Sie dies jetzt nachholen. Die Kenntnis der internen Vorgänge ist zum Verständnis dieses Abschnittes notwendig.

In unserem Beispiel eines Musikstücks für „Klavier“ und „Streichorchester“ würde die MidiKanal-Zuordnung lauten:

```
MIDIKANAL
     1 -> 1-8
     2 -> 9-16
```

Dies bedeutet, dass alle NOTE-ON- bzw. NOTE-OFF-Meldungen, die auf MIDI-Kanal 1 empfangen werden, an das „Klavier“ weitergeleitet werden und alle auf Kanal 2 empfangenen an das „Streichorchester“. Grundvoraussetzung hierfür ist, dass Sie an Ihrem Synthesizer/Sampler den Multi-Mode richtig eingestellt haben und die Pitchbend-Range bei jeden Instrument einen Halbton beträgt.

Das Besondere bei der Benutzung mehrerer MidiKanäle ist, daß MUTABOR bei dieser Aufspaltung jeden MidiKanal als *völlig eigenständig* behandelt. Verschiedene MidiKanäle können zur gleichen Zeit unterschiedliche Stimmungslogiken oder Tonsysteme spielen.

Auf dem Bildschirm wird aber nur eines der Instrumente/Kanäle angezeigt. Welches das ist, kann mit den Zifferntasten 1–9 gewählt werden. Auf diesen angezeigten MidiKanal (Instrument) bezieht sich auch der Auslöser `Taste`.

```
LOGIK
    Hans  Taste h = drittel_ton [ ]
    Meier Taste m = halb_ton [ ]
MIDIKANAL
    1 -> 1-8    "Klavier"
    2 -> 9-16   "Streicher"
```

Wenn das Klavier in der Logik Hans spielen soll, so drückt man zunächst die Tasten ‘1’ und ‘h’. Wenn dann die Streicher in der Logik Meier spielen sollen, so drückt man die Tasten ‘2’ und ‘m’. Dabei wechselt man zunächst zum MidiKanal 2 und aktiviert dann die Logik Meier. Der Wechsel zum MidiKanal 2 bleibt bestehen, so dass Sie nicht vor jeder einzelnen Umstimmung den MidiKanal *vorwählen* müssen. Da bisher keine MidiKanal-Deklaration benutzt wurde, gab es nur den MidiKanal 1 und man brauchte nie eine Ziffer *vorzuwählen*. Natürlich kann durch eine erneute Zifferntaste zum nächsten MidiKanal gewechselt werden. Es sind aber nur so viele MidiKanäle vorhanden, wie in dem Logik-Programm deklariert wurden. Auf MidiKanäle, die nicht deklariert wurden, reagiert MUTABOR einfach nicht.

Die MidiKanäle müssen nicht fortlaufend durchnummeriert werden. Man kann z. B. auch deklarieren:

`MIDIKANAL 4 -> 1-16`

und schon reagiert MUTABOR *nur* auf den MIDI-Kanal 4. Dies ist für die Zusammenschaltung mehrerer MUTABOR bzw. mehrerer Synthesizer in einer MIDI-Kette wichtig.

16. Kommentare

Wie in dem vorherigen Programmierbeispiel zu sehen war, kann man so genannte *Kommentare* in ein Programm hineinschreiben. Kommentare dienen dazu, Bemerkungen und Gedanken zum Programm festzuhalten. Sie sind nur für den menschlichen Leser von Bedeutung und beeinflussen in keinster Weise das Logikprogramm. Kommentare sind dadurch gekennzeichnet, dass sie in doppelten Anführungszeichen (") stehen. Ein Kommentar darf seinerseits keine Anführungszeichen enthalten, aber durchaus über mehrere Zeilen gehen. Beispiel:

```
INTERVALL
      Drittelton = 18 Wurzel 2  "Teilt die Oktave in
                                 achtzehn gleiche Teile"
TON
      a = 440 "Der Kammerton wird auf 440Hz gesetzt"

TONSYSTEM
      Drittel = 69 [ a ] Drittelton
         "Das Tonsystem Drittel beginnt beim
          eingestrichenen a' (=69), die FT besteht aus
          einem Ton (dem Ton a=440Hz) und diese Struktur
          wird um einen Drittelton verschoben wiederholt."

LOGIK
      Drittel Taste D = Drittel [ ]
         "Die nicht-mutierende, statische Logik Drittel wird
          durch Druck der Taste D aktiviert."
```

Dies ist ein Musterbeispiel für gut dokumentiertes Programmieren. Mit Hilfe von Kommentaren im Logikprogrammtext ist es auch einem anderen Anwender möglich, zu verstehen, wie die Stimmungslogik funktioniert – selbst wenn dieser noch nie Stimmungslogiken unter Mutabor programmiert hat.

Eine weitere Verwendungsmöglichkeit für Kommentare ist das „Auskommentieren“ von Programmteilen, die nicht oder noch nicht übersetzt werden sollen. Wenn die konzeptuelle Planung eines Logikprogramms bereits so weit fortgeschritten ist, dass die Logiken fertig programmiert sind, aber noch nicht die von ihnen benutzten Tonsysteme und Umstimmungen, z. B. das Logikprogramm

```
LOGIK
     Garten Taste G = Gras [
```

```
                             Blume       -> Strauch(4)
                             Kanne       -> Gras
                             FORM Akkord -> Umstimmaktion(ABSTAND)
                          ]
TONSYSTEM
     Gras = 60 [ c ] Murksintervall
TON
     c = 266
INTERVALL
     Murksintervall = 1.088463542657 : 1
```

würde vom Compiler nicht übersetzt werden können, da in der Logik `Garten` die Auslöser `Blume`, `Kanne` und `Akkord`, sowie die Aktionen `Strauch` und `Umstimmaktion` noch nicht deklariert worden sind.

Wenn Sie aber den bereits fertig programmierten Bestandteil des Logikprogramms testen möchten, ohne das in der Logik formulierte Konzept zu löschen, so müssen Sie bloß die noch nicht deklarierten Bezeichner „auskommentieren“:

```
LOGIK
     Garten Taste G = Gras [
  "
                       Blume       -> Strauch(4)
                       Kanne       -> Gras
                       FORM Akkord -> Umstimmaktion(ABSTAND)
  "
                           ]
TONSYSTEM
     Gras = 60 [ c ] Murksintervall
TON
     c = 266
INTERVALL
     Murksintervall = 1.088463542657 : 1
```

Der gesamte Text, welcher *zwischen* den Anführungszeichen steht wird vom Compiler als Kommentar behandelt und nicht weiter beachtet. Dadurch sind die Bezeichner `Blume`, `Kanne`, `Akkord`, `Strauch` und `Umstimmaktion` nicht benutzt und werden vom Compiler auch nicht als „nicht deklariert“ bemängelt. Das Programm mit den auskommentierten Stellen kann jetzt fehlerfrei übersetzt werden.

Dies ist sehr wichtig, wenn Sie Ihre Programme auf der höchsten Ebene zu schreiben beginnen (also mit den Logiken) und Objekte benutzen, die Sie noch nicht deklariert haben. Wenn Sie die entsprechenden Stellen auskommentieren, so können Sie das Logikprogramm-Fragment trotzdem austesten.

A. Zusammenfassung aller Elemente der Programmiersprache

Dieses Kapitel gibt Ihnen noch einmal eine vollständige Zusammenfassung aller programmtechnischer Konstruktionen, mit denen Sie MUTABOR Mikrotöne entlocken können. Sie werden hier vielem Bekannten begegnen, aber auch Neues entdecken. Da Sie bereits mit dem Konzept und den meisten Elementen der Programmiersprache von MUTABOR vertraut sind, werden die grundlegenden Konstruktionen nicht noch einmal im Detail erläutert. Sie sollten also wissen, was unter einer Tondeklaration, einer Umstimmung oder einer MIDI-Nachricht, ... zu verstehen ist. Wir wollen Sie in diesem Kapitel auf mögliche Konstruktionen aufmerksam machen, die die Flexibilität und Universalität von MUTABOR hervorstellen und dazu dienen sollen, *kreative Anregungen* für das Experimentieren mit ausgefallenen Stimmungslogiken zu geben.

Die vollständige Definition der Syntax der Programmiersprache finden Sie im Referenzhandbuch, dort jedoch in der abstrakten Form der Syntaxdiagramme. Um Ihnen also noch einmal eine „letzte Chance" zu geben, die Programmiersprache durch das Beschauen von Beispiel zu lernen, wurde dieses Kapitel konzipiert.

A.1. Intervalle und Töne

Ein Intervall ist bei MUTABOR ein Frequenzverhältnis und kann als Zahlenverhältnis angegeben werden.

```
INTERVALL  meier = 5 : 4
```

Wenn das Intervall sozusagen „in die andere Richtung" weisen soll, so braucht man nur das Zahlenverhältnis umzukehren:

```
INTERVALL umgekehrt = 4 : 5
```

Wenn ein Intervall der soundsovielte Teil eines anderen Intervalles sein soll, so muss man wegen dem logarithmischen Gehör die entsprechende Wurzel ziehen. Wenn also die Oktave in genau zwölf gleichgroße Teile geteilt werden soll, so muss man die zwölfte Wurzel ziehen:

```
INTERVALL halbton = 12 WURZEL 2
```

Natürlich ist es auch möglich, sich ein beliebiges Phantasie-Intervall zu definieren. Wenn dies z. B. die Kommazahl 3.14159 sein soll, so schreibt man einfach

```
INTERVALL phantasie = 3.14159 : 1
```

Mit dem bisherigen kann man Intervalle definieren, die einfach einen festen Wert haben. Des weiteren kann man Intervalle auch aus anderen Intervallen zusammensetzen.

```
INTERVALL seltsam = phantasie - 3 halbton + 2 cent
```

Dieses Intervall-Zusammensetzen muss in sich konsistent sein. So kann man nicht zwei Intervalle gegenseitig voneinander abhängen lassen.

```
INTERVALL eins = phantasie + zwei + 5.1 cent
          zwei = eins - 3 halbton
```

Die Intervalle `eins` und `zwei` sind in dieser Konstruktion nicht eindeutig definiert und deshalb werden solche Kreis-Bezüglichkeiten überprüft.

Kommen wir nun zu den Tönen: Ein Ton ist bei MUTABOR ein Name oder Bezeichner, der letztendlich eine Frequenz repräsentiert. Mit welchem Instrument oder in welcher Klangfarbe die Töne gespielt werden, ist Sache des angeschlossenen Synthesizers, hat also mit MUTABOR nichts zu tun. Hier deklarieren wir, dass ein Ton eine bestimmte Frequenz hat:

```
TON c   = 130.81
    d   = 146.84
    e   = 164.8
    f   = 174.6
    g   = 196
    a   = 220
    h   = 246.9
```

Dasselbe hätten wir mit der folgenden Deklaration erreicht:

```
INTERVALL halbton = 12 WURZEL 2

TON c   = a - 9 halbton
    d   = a - 7 halbton
    e   = a - 5 halbton
    f   = a - 4 halbton
    g   = a - 2 halbton
    a   = 220
    h   = a + 2 halbton
```

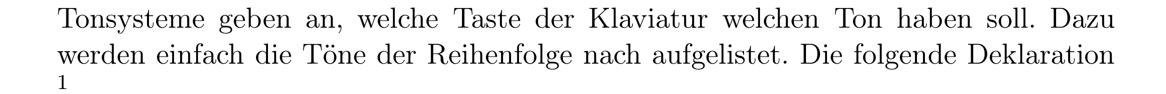
Man sieht also, dass Töne sowohl eine feste Frequenz haben können, als auch relativ zu anderen Tönen definiert werden können. Die relativ definierten Töne werden von MUTABOR berechnet, so dass auch diese Töne letztendlich eine feste Frequenz haben. Die relativ angegebenen Töne können beliebig komplex sein:

```
TON otto = c + 2 seltsam - 4 phantasie + 3 halbton + 0.2 cent
```

Es muss nur gewährleistet sein, dass die Töne nicht gegenseitig voneinander abhängen und dass die benutzten Bezugstöne und Intervalle korrekt deklariert werden. Es gibt also eine Menge Möglichkeiten, um sich Töne zusammenzustellen.

A.2. Tonsysteme

Tonsysteme geben an, welche Taste der Klaviatur welchen Ton haben soll. Dazu werden einfach die Töne der Reihenfolge nach aufgelistet. Die folgende Deklaration [1]



```
TONSYSTEM gleichstufig = 48 [c, ,d, ,e,f, ,g, ,a, ,h] oktave
```

wird im folgenden nochmal geschrieben, aber so, dass die einzelnen Elemente mit Kommentaren versehen sind. Diese Schreibweise wird von MUTABOR genauso verstanden, wie obiges Beispiel:

[1] Beispielprogramm `komentar.mut`

```
TONSYSTEM     ”dieses Wort bestimmt, dass jetzt”
              ”Tonsystem-Deklarationen folgen”

gleichstufig  ”das ist der Name des Tonsystems”

=             ”ein Gleichheitszeichen, damit sichtbar”
              ”ist, dass hier etwas definiert wird”

48            ”das ist die MIDI-Nummer der ersten Taste,”
              ”die hier einen Ton zugeordnet bekommt”

[             ”Die Liste der Töne beginnt”

c             ”Taste 48 bekommt Ton c”
,             ”nächster Ton folgt”
              ”Taste 49 ist stumm”
,             ”nächster Ton folgt”
d             ”Taste 50 bekommt Ton d”
,             ”nächster Ton folgt”
              ”Taste 51 ist stumm”
,             ”nächster Ton folgt”
e             ”Taste 52 bekommt Ton e”
,             ”nächster Ton folgt”
f             ”Taste 53 bekommt Ton f”
,             ”nächster Ton folgt”
              ”Taste 54 ist stumm”
,             ”nächster Ton folgt”
g             ”Taste 55 bekommt Ton g”
,             ”nächster Ton folgt”
              ”Taste 56 ist stumm”
,             ”nächster Ton folgt”
a             ”Taste 57 bekommt Ton a”
,             ”nächster Ton folgt”
              ”Taste 58 ist stumm”
,             ”nächster Ton folgt”
h             ”Taste 59 bekommt Ton h”

]             ”die Liste der Töne ist zu Ende”

oktave        ”die Tasten 48 bis 59 wiederholen sich”
              ”nach beiden Seiten bis zum Anschlag”
              ”und verschieben die Tonhöhe dabei”
              ”um jeweils eine Oktave”
```

Man sieht hier, wie die einzelnen Töne den Tasten des Keyboards zugeordnet werden. Nach rechts geht das Schema folgendermaßen weiter:

Taste 60 → c + oktave
Taste 61 → Stumm
Taste 62 → d + oktave
Taste 63 → Stumm
Taste 64 → e + oktave
+ ... → ...
Taste 72 → c + 2 oktave
Taste 73 → Stumm
Taste 74 → d + 2 oktave
... → ...
Taste 84 → c + 3 oktave

und nach links geht das Schema auch entsprechend weiter. Wenn man die Einschränkung fallen lassen will, dass sich das Muster der Töne periodisch wiederholt, so braucht man nur das Tonsystem so breit zu wählen, dass es die ganze Klaviatur abdeckt.

```
TONSYSTEM sehr_breit = 36 [ ton36,ton37,ton38, ... ,ton96 ] dummy
```

In diesem Fall kann als Periodenintervall (`dummy`) ein beliebiges Intervall eingesetzt werden, da es im Bereich der verfügbaren Klaviaturtasten sowieso keine Auswirkungen hat. Man muss jetzt 61 Töne[2] (`ton36`–`ton96`) einzeln deklarieren, aber dafür hat man die Tonhöhe jeder einzelnen Taste in der Hand. Man kann auch das Tonsystem kürzer wählen und z. B. ein Muster aus 3 Tasten wiederholen lassen.

```
TONSYSTEM drei = 50 [ t1, t2, t3 ] halbton
TON t1 = t2 - 10 cent
    t2 = 440
    t3 = t2 + 10 cent
INTERVALL halbton = 12 wurzel 2
          cent = 1200 wurzel 2
```

Jetzt liegt der Kammerton a' auf der Taste 51 (=es) und die weiteren Halbtöne sind nicht direkt daneben, sondern jeweils drei Tasten weiter. Dafür ist jeder Ton sozusagen “eingerahmt“ von zwei weiteren Tönen, die um ±10 Cent verstimmt sind.

A.3. Umstimmungen und Verwandte

Eine Umstimmung verändert das aktuelle Tonsystem, indem einzelne Parameter der Fundamentaltonleiter geändert werden. So kann man die Verankerungstaste verschieben, die Breite oder das Periodenintervall ändern und einzelne Töne ändern. Dies sind die vier Grundmöglichkeiten, um ein Tonsystem zu ändern.

[2]Vorausgesetzt, dass ein Keyboard mit 61 Tasten benutzt wird.

```
UMSTIMMUNG
   u1 = 51 [ ]                "Setzt Verankerungstaste neu"
   u2 = [ << 5 >> ]           "Setzt die Breite auf 5"
   u3 = [ << @ + 1 >> ]       "Verbreitert um einen Ton"
   u4 = [ @, xy, @ ]          "Setzt den zweiten Ton neu"
   u5 = [ @, @, @ - 3 cent ] "Erniedrigt den dritten Ton"
   u6 = [ ] ganzton - 5 cent "Setzt das Periodenintervall neu"
   u7 = [ ] @ + halbton       "Vergroessert das Periodenintervall"
```

Da eine einzelne Umstimmung im allgemeinen nicht ausreichend ist, kann man auch mehrere Umstimmungen auf einmal durchführen.

```
UMSTIMMUNG mehrfach = { u1, u3, u7 }
```

Wenn jetzt die Umstimmung `mehrfach` durchgeführt wird, so werden der Reihe nach die Umstimmungen `u1, u3` und `u7` durchgeführt. Sollte zusätzlich zu den drei Umstimmungen noch eine Nachricht an den Synthesizer geschickt werden, so schreibt man:

```
UMSTIMMUNG mehrfach = { u1, u3, u7, MidiOut (#C0, 17, 22) }
```

und schon bekommt der Synthesizer eine drei Bytes lange MIDI-Nachricht geschickt, sobald die Umstimmung `mehrfach` durchgeführt wird. Bei der Angabe von Verankerungstaste und Breite darf keine Kommazahl sondern nur eine ganze Zahl angegeben werden. Diese Zahl muss aber nicht konstant sein, sondern sie kann der Umstimmung auch als Parameter mitgegeben werden.

```
UMSTIMMUNG  setze_anker (x) = x [ ]
```

Diese Umstimmung kann nun von verschiedenen Stellen aus mit unterschiedlichen Parametern aufgerufen werden.

```
setze_anker (48)
setze_anker (36)
setze_anker (59)
```

Eine Umstimmung kann auch mehrere Parameter haben, und diese Parameter an andere Umstimmungen weitergeben.

```
UMSTIMMUNG meier (x,y,z) = {   setze_anker (z),
                               setze_breite (x),
                               verschiebe (y),
                               midiout (#D0, 17, 4)
                           }

     setze_breite (x) = [ << x >> ]

     verschiebe (j) = @ + j [ ]
```

Es ist auch möglich, in Abhängigkeit von einem Parameter eine Fallunterscheidung zu treffen. Hier soll die Verankerungstaste nur dann verschoben werden, wenn der Parameter *ungerade* ist.

```
UMSTIMMUNG ungerade (x) = x {
                              -5 -> verschiebe (x)
                              -3 -> verschiebe (x)
                              -1 -> verschiebe (x)
                               1 -> verschiebe (x)
                               3 -> verschiebe (x)
                               5 -> verschiebe (x)
                               7 -> verschiebe (x)
                               9 -> verschiebe (x)
                             }
```

Man kann auch je nach Parameter verschiedene andere Umstimmungen oder MIDI-Nachrichten auslösen:

```
UMSTIMMUNG super (x) = x {
                          0 -> u1 , midiout (#c0, 0, 0)
                          1 -> u2, u3, verschiebe (4)
                          2 -> midiout (#b0, 2, 5), u4
                  ansonsten -> midiout (#b0, 0, 127)
             }
```

Es ist allerdings verboten, dass Umstimmungen sich gegenseitig aufrufen. Das würde zu einem unendlichen PingPong von Umstimmungen führen und ist nicht sinnvoll. Sinnvoll ist jedoch, dass eine Umstimmung ein komplettes Tonsystem aufruft. In diesem Fall wird das angegebene Tonsystem eingestellt. Es ist sogar möglich in einer Umstimmung eine andere Logik aufzurufen. In diesem Fall wird in diese Logik gewechselt und, falls vorhanden, deren Einstimmung durchgeführt.

A.4. Harmonien und Logiken

Eine Logik besteht aus einer Einstimmung und einer Liste von Stimmungsregeln. Die Regeln wiederum geben an, was bei welchem Ereignis zu geschehen hat.

```
LOGIK einfach  TASTE e = drei [ ]
```

Die Logik `einfach` wird aktiviert, wenn man auf der Computertastatur die Taste E drückt und hat als Einstimmung das Tonsystem `drei`. Weiteres passiert in dieser Logik nicht. Wenn man jedoch einige Anweisungen in die eckigen Klammern schreibt, so arbeitet die Logik.

```
LOGIK einfach  TASTE e = drei [
                              taste x -> u1
```

```
                                  taste y -> u3 , u5
                                  taste z -> midiout (#C0, 1)
]
```

Jetzt kann man mit der Taste E die Logik `einfach` aufrufen und dann mit den Tasten `x, y` und `z` die entsprechenden Umstimmungen oder das MidiOut aufrufen. Aber Tasten sind nicht die einzigen Ereignisse, auf die eine Logik reagieren kann.

```
LOGIK program_change  TASTE p = drei [
                           midiin (#C0, 05) -> u1
                           midiin (#C0, 09) -> u3 , u5
                           midiin (#C0, 22) -> midiout (#C0, 1)
                      ]
```

Die Logik `program_change` wird mit der Taste P aktiviert und reagiert dann auf die drei angegebenen MIDI-Nachrichten. Es können auch gemischte Ereignisse von einer Logik bearbeitet werden:

```
LOGIK mehrfach  TASTE m =  [
                     taste x          -> u1
                     taste y          -> u3 , u5
                     taste z          -> midiout (#C0, 1)
                     midiin (#C0, 05) -> u4
                     midiin (#C0, 09) -> u3 , u2
                     midiin (#C0, 22) -> midiout (#B0, 127, 4)
                ]
```

Die Logik `mehrfach` wird mit der Taste M aktiviert und reagiert dann auf drei Tasten und drei MIDI-Nachrichten. Ferner enthält sie keine Einstimmung. Es bleibt bei ihrem Aufruf also die bisherige Stimmung bestehen. Die dritte Möglichkeit, in einer Logik auf etwas zu reagieren, ist die Harmonie-Analyse.

A.4.1. Komplizierte Fälle

Kompliziertere Auslöser als Computer-Tasten oder MIDI-Nachrichten sind Harmonien und Harmonieformen. (Eigentlich handelt es sich hierbei um "Tastenmuster„, denn eine Harmonieanalyse im Sinne der Musikwissenschaft ist mit Mutabor noch nicht möglich.) Hier wird eine Harmonie als Menge von gedrückten Tasten in der Projektionstonleiter verstanden. Beim Erkennen der angegebenen Harmonie gilt der Auslöser als erfüllt. Beispiel:

```
HARMONIE
      c_dur   = {0, 4, 7}
      cis_dur = {1, 5, 8}
      d_dur   = {2, 6, 9}
      dis_dur = {3, 7, 10}
```

```
      e_dur   = {4, 8, 11}

LOGIK mutiere_dur Taste M = [
             c_dur   -> u1
             cis_dur -> u2
             d_dur   -> u3
             dis_dur -> u4
             e_dur   -> u5
           ansonsten -> u99
         ]
```

Sobald die Logik `mutiere_dur` aktiviert ist, wird bei jeder der fünf angegebenen Harmonien eine Umstimmung durchgeführt. Falls in einem Moment mal keine der Harmonien zutrifft (sondern ein anderes Tastenmuster gedrückt ist) wird die Umstimmung u99 durchgeführt.

Man kann auch eine Harmonie-Form Analyse durchführen, bei der es nicht auf die Verschiebung der Harmonie bezüglich des Grundtons ankommt:

```
HARMONIE
      dur   = {0, 4, 7}

LOGIK mutiere_dur_2 Taste M = [
             FORM dur   -> umst (ABSTAND)
             ansonsten  -> u99
         ]
```

Hier wird jede Dur-Harmonieform erkannt, und zwar nicht nur die fünf Harmonien aus dem vorigen Beispiel, sondern alle zwölf Harmonieformen – vorausgesetzt, die Breite des Tonsystems ist zwölf. In dem Parameter `ABSTAND` wird als Nummer mitgegeben, um wieviele Tastenschritte die erkannte Harmonie vom Prototyp der Harmonieform entfernt ist, ABSTAND liefert also die Stufe, auf der die erkannte Harmonie bezüglich dem aktuellen Zentrum (=Verankerungstaste) steht. Man könnte dies für eine *Auswählende Umstimmung* verwenden und die Umstimmung `umst` definieren als:

```
UMSTIMMUNG
    umst (x) = x {
                   0 -> u1  "Fall c_dur"
                   1 -> u2  "Fall cis_dur"
                   2 -> u3  "Fall d_dur"
                   3 -> u4  "Fall dis_dur"
                   4 -> u5  "Fall e_dur"
             }
```

Möchte man zusätzlich erreichen, dass nicht jeder `Dur`-Dreiklang erkannt wird, so kann man noch folgende Einschränkungen machen:

```
HARMONIE
      dur   = {0, 4, 7}

LOGIK mutiere_dur_3 Taste M = [
             FORM 4 ~ dur  -> umst (ABSTAND)
         ]
```

Jetzt wird die Harmonie-Form Analyse genauso durchgeführt, aber der tiefste Ton muss vom Tonigkeitstyp 4 sein. Ebenso kann einschränkend gesagt werden, dass der höchste Ton von einer bestimmten Art sein soll:

```
HARMONIE
      dur   = {0, 4, 7}

LOGIK mutiere_dur_3 Taste M = [
             FORM dur ~ 7  -> umst (ABSTAND)
         ]
```

Jetzt reagiert der Auslöser `dur ~ 7` nur dann, wenn der höchste Ton ein g ist. (Bei zwölfstufiger Fundamentaltonleiter). Es sind auch beide Einschränkungen gleichzeitig zugelassen. Durch diese einschränkende Analyse können verschiedene Lagen und Umkehrungen derselben Harmonieform unterschiedliche Umstimmungen auslösen.

B. Änderungen der Sprachsyntax von Version 2.0 zu Version 2.1 und Version 3.0

Leider ließen sich folgende Konstruktionen nicht „aufwärtskompatibel“ herstellen:

- In Intervalldeklarationen sind reine Ganz- oder Kommazahlen nicht mehr erlaubt. Die Intervalldeklarationen `INTERVALL Oktave = 2` oder `Halbton = 1.05946` ist ab Version 2.1 syntaktisch falsch. Nichtsdestotrotz können Sie solche Intervalle erzeugen, indem Sie sie als Verhältnis zu eins darstellen: `INTERVALL Oktave = 2:1` und `Halbton = 1.05946:1`.

- Die Deklaration von INSTRUMENT ist durch die Deklaration von MIDIKANAL ersetzt worden. Die Bedeutung hat sich jedoch nicht geändert.

- An verschiedenen Stellen wo bisher nur ein einzelnes Intervall zulässig war, kann jetzt auch eine Kombination von mehreren Intervallen angegeben werden. Diese Fälle sind:

  Intervalldeklaration `Intervall kombi = 2 Terz - Oktave + 4 cent`

  Tonsystemdeklaration `Tonsystem Meier = 69 [ a ] Halbton + 0.5 cent`

  Umstimmungsdeklaration `Umstimmung Otto_absolut = [ ] Halbton + 0.7 cent`

  `Umstimmung Otto_relativ = [ ] @ + 3 cent`

Falls also Logikprogramme, die Sie unter der Version 2.0 erfolgreich geschrieben haben, bei neueren Versionen plötzlich Syntaxfehler erzeugen, so liegt die Ursache mit Sicherheit an einer dieser Nicht-Aufwärtskompatibilitäten. Es sei noch einmal betont, dass sich durch diese Änderung der Sprachsyntax keinerlei Einschränkungen für die Funktionalität oder die Leistungsfähigkeit von MUTABOR ergeben haben.

Eine weitere Änderung ab der Version 2.1 ist, dass zu hohe oder zu tiefe Frequenzen nicht mehr unterdrückt werden, sondern in den „hörbaren Bereich“ hineinprojiziert werden. Näheres siehe Abschnitt „Handhabung von Grenzfällen“.

C. Fehlerursachen

C.1. Fehlermeldungen des Compilers

In diesem Anhang werden sämtliche Fehlermeldungen des Compilers aufgeführt und erläutert, wie sie entstehen und wie sie zu beheben sind. Anstelle der drei Fragezeichen (???) erscheint der fehlerhafte Wert oder ein Hinweis auf die Stelle wo sich der Fehler befindet.

C.1.1. Allgemeine Fehler

Datei- und Systemfehler sollten eigentlich nie auftreten. Sollte MUTABOR trotzdem unter **reproduzierbaren** Umständen abstürzen, so informieren Sie uns, damit wir den Fehler beseitigen können.

Undefinierter Fehler in ??? ist ein Fehler, der auf Programmierfehlern innerhalb von MUTABOR beruht. Falls dieser Fehler auftreten sollte, so prüfen Sie bitte, unter welchen *reproduzierbaren* Bedingungen der Fehler erscheint und melden uns diesen Fehler. Wir sind bemüht, solche Fehler umgehend zu beheben.

Syntaktischer Fehler, nicht näher spezifiziert. (Zeile ???) tritt auf, wenn das Logik-Programm sich in gravierender Weise nicht an die vorgegebene Syntax hält. Zur Behebung des Fehlers lesen Sie bitte im Referenzhandbuch die Syntaxgraphen. Es kann sich bei diesem Fehler um vergessene Kommata, um falsche Namensschreibungen, um falsche Wahl der Klammern, um Vergessen von Zeichen, ... – kurz: um so ziemlich alles handeln, was der Compiler nicht versteht.

Ungültiges Zeichen im Quelltext: asc=??? (Zeile ???) tritt auf, wenn in der Datei, die das Logik-Programm enthält, Zeichen enthalten sind, die nicht zum eigentlichen Text gehören. Manche Editoren schreiben zusätzliche Steuerzeichen in den Text, um z. B. Absätze, Fettschrift ... zu markieren. Solche Steuerzeichen stören das Logik-Programm und müssen entfernt werden. Der Text des Logik-Programms muss also im ASCII-Format gespeichert sein. Als Fehlercode wird der Dezimalwert des fehlerhaften Zeichens und die Zeilen-Nummer angezeigt, in der das fehlerhafte Zeichen auftrat.

Kann Datei nicht öffnen: ??? tritt auf, wenn die angegebene Datei nicht existiert, oder fehlerhaft ist. Es kann auch sein, dass die Diskette voll ist oder dass die Datei schreibgeschützt ist.

Speichermangel ! tritt auf, wenn der Speicher Ihres Computers nicht mehr ausreicht. Es ist prinzipiell möglich, dass einfach das Logik-Programm zu komplex ist. Wahrscheinlicher ist jedoch, dass der Speicher von anderen Programmen mitbenutzt wird.

C.1.2. Doppeldeklarationen

treten als Fehler auf, wenn mehrere Objekte gleichen Typs und gleichen Namens deklariert werden. Wenn man diese Namen benutzen würde, ist nicht mehr eindeutig, welche Deklaration gemeint ist.

Der Intervallname ??? wurde doppelt benutzt

Der Tonname ??? wurde doppelt benutzt

Der Tonsystemname ??? wurde doppelt benutzt

Der Umstimmungsname ??? wurde doppelt benutzt

Der Harmoniename ??? wurde doppelt benutzt

Der Logikname ??? wurde doppelt benutzt

Der Parametername ??? wurde doppelt benutzt

Das MIDI-Instrument ??? wurde doppelt benutzt

Überlappungsfehler bei der MIDI-Zuordnung: Kanal ??? wurde doppelt benutzt

Mehr als ein ANSONSTEN-Auslöser in Logik ???

Mehr als ein ANSONSTEN in Umstimmung ???

Die Alternative ??? ist doppelt in ???

C.1.3. Undefinierte Symbole

treten als Fehler auf, wenn man etwas benutzt, ohne deklariert zu haben, um was es sich handeln soll. Zur Fehlerbehebung kann man die Deklaration des Symbols nachholen oder die Benutzung des Symbols ausschließen, z. B. indem man den Programmteil, der das undefinierte Symbol benutzt, in Kommentar (") einschließt.

Unbekanntes Symbol: ???

Unbekanntes Intervall: ???

Unbekannter Ton: ???

Unbekanntes Tonsystem: ???

Unbekannte Umstimmung: ???

Unbekannte Harmonie: ??? (in Logik ???)

Unbekannter Parametername: ??? in ???

Unbekannte Periode: ??? (in Tonsystem ???)

Unbekannter Ton: ??? (in Tonsystem ???)

Unbekannter Ton: ??? (in Umstimmung ???)

Unbekanntes Intervall: ??? (in Umstimmung ???)

Unbekannte Umstimmung: ??? (in Umstimmung ???)

Unbekannte Einstimmung: ??? (in Logik ???)

Unbekannte Aktion: ??? (in Logik ???)

C.1.4. Bereichsüber- bzw. -Unterschreitungen

treten auf, wenn der angegebene Wert nicht in den erlaubten Grenzen liegt.

MIDI-Kanal ungültig (zulässig: 1 bis 16)

Unzulässiger Wert von Intervall ???

Taste ??? liegt außerhalb des Wertebereichs 36..96 (in Tonsystem ???)

In Umstimmung ??? ist der Ton ??? nicht komplex

Unzulässiger Wert in ???

Unzulässiger MIDI-Code in Logik ??? (erwarte ???)

Logik ??? darf nicht mit ANSONSTEN aufgerufen werden

Als Taste eines Auslösers wurde kein einzelner Buchstabe angegeben TASTE ???

C.1.5. Parameterfehler

treten auf, wenn der Mechanismus zur Übergabe von Parametern innerhalb von Aufrufen oder Umstimmungen nicht passt, oder wenn Parameter benutzt werden, die nicht ordnungsgemäß deklariert wurden. Prüfen Sie in diesem Fall die gemeldete Fehlerstelle und die Deklaration der Umstimmung.

In Umstimmungsbund/case ??? stimmen in Umstimmung ??? die Parameter nicht

In Logik ??? stimmt die Parameter- anzahl von ??? nicht

In Logik ??? ist die Einstimmung ??? nicht parameterlos

in Umstimmungsbund ??? ist das Tonsystem ??? nicht parameterlos

in Umstimmungs_case ??? ist das Tonsystem ??? nicht parameterlos

C.1.6. Gegenseitige Abhängigkeiten

sind innerhalb von Tönen und Umstimmungen möglich, aber verboten. Stellen Sie die Struktur der Abhängigkeiten so um, dass gegenseitige Abhängigkeiten nicht mehr vorkommen. Die Struktur der Abhängigkeiten muss einen gerichteten, azyklischen Graphen ergeben, der nicht notwendigerweise zusammenhängend ist.[1]

Die Töne ??? und ??? hängen gegenseitig voneinander ab

Die Umstimmungen/Logiken ??? und ??? hängen gegenseitig voneinander ab

Die Intervalle ??? und ??? hängen gegenseitig voneinander ab

[1] Zu Deutsch: zeichnet man sich die Abhängigkeiten im Programm folgendermaßen auf: Die Töne, Intervalle usw. werden als Punkte dargestellt. Immer dann, wenn ein Punkt (A) in der Deklaration eines anderen Punktes (B) auftaucht, zeichnen wir einen Pfeil von B nach A. Diese Zeichnung ergibt am Ende einen gerichteten Graphen. Finden wir einen Punkt C, von dem ein Pfeil weggeht und man entlang der Pfeile auch wieder zurückkommt, so ist das ein Kreis. Der Graph ist dann nicht azyklisch. Es darf also keinen solchen Kreis geben.

C.1.7. Syntaxfehler

treten auf, wenn das Logik-Programm nicht den Syntaxgraphen des Referenzhandbuchs entspricht. Im Gegensatz zu *Syntaktischer Fehler, nicht näher spezifiziert. (Zeile ???)* wird hier Bezug auf das gerade gelesene Programmkonstrukt genommen.

Falsches Zeichen! Erwarte ??? (Zeile ???)

Fehlerhafte Intervall- deklaration in Intervall ???

Fehlerhafte Tondeklaration von Ton ???

Fehlerhafte Tonsystem- deklaration. (Zeile ???)

Ungültige Pameterliste in Zeile ???

Ungültige Umstimmung in Zeile ???

Ungültige Harmoniedekl. bei ???

Fehlerhafte MIDI-Liste in Zeile ???

Nach einem # folgt keine Hex-Ziffer (Zeile ???)

C.2. Warnungen

sind Meldungen an den Benutzer, dass das Programm zwar korrekt funktioniert, aber zweifelhafte Konstruktionen enthält. Es ist nützlich, diese Warnungen zu beachten, es sei denn, man beabsichtigt gerade die dadurch hervorgerufenen Effekte.

MIDI-Auslöser in Logik ??? beginnt nicht mit Kanal 0- Statusbyte. Wert wurde korrigiert! Siehe Abschnitt 12.3

Unmöglicher Harmonieauslöser in Logik ??? Siehe Abschnitt 7.6

Mehrdeutiger Auslöser ??? in Logik ???

Auslöser ANSONSTEN ist nicht der letzte in Logik ??? Diese Warnung ist belanglos, es dient aber der besseren Lesbarkeit, wenn diese Alternative am Ende und nicht irgendwo zwischen den Alternativen steht.

Undefinierte Compilerwarnung ist eine Warnung, die auf Programmierfehlern innerhalb von Mutabor beruht. Falls diese Warnung auftreten sollte, so prüfen Sie bitte, unter welchen **reproduzierbaren** Bedingungen die Warnung erscheint und melden uns diesen Fehler. Wir sind bemüht, solche Fehler umgehend zu beheben.

Verschwendung bei Kanalzuordnung bedeutet, dass Sie bei der MIDI-Kanal-Deklaration nicht alle 16 möglichen MIDI-Kanäle benutzen.

Konfigurationsdatei fehlerhaft In der Datei `MUTABOR.CFG` sind fehlerhafte Einträge. Sie können diese Datei löschen und danach Mutabor erneut starten und wieder verlassen. Dabei wird die Datei mit den Grundeinstellungen neu erzeugt.

C.3. Sonstige Fehler

C.3.1. Es ist nichts zu hören

Wenn Sie nichts hören, so kann das vielfältige Ursachen haben. Solche Pannen sind natürlich ärgerlich, und wir sind deswegen bemüht, hier eine möglichst vollständige Liste an Ursachen anzugeben, die schuld daran sein könnten, dass Sie nichts hören.

- Sind alle Geräte angeschaltet und mit Strom versorgt ? Ist der Verstärker richtig eingestellt und die Lautstärke aufgedreht ? ... Auch wenn es lächerlich erscheint — dieser Fehler ist erfahrungsgemäß der häufigste.
- Das Masterkeyboard sendet auf einem MIDI-Kanal, welcher nicht als MidiKanal bei Mutabor eingetragen ist. Gewöhnlicherweise sollte das Masterkeyboard auf MIDI-Kanal 1 senden.
- Die Klangerzeugungseinheit (Synthesizer/Sampler) ist nicht korrekt für den Betrieb mit Mutabor eingestellt. Es muss auf jedem MIDI-Kanal die gleiche Klangfarbe mit Pitch-Bend-Range 1 eingestellt sein (Multi-Mode).
- Es ist eine Stimmungslogik aktiv, die nur aus stumm geschalteten Tasten besteht, oder aus Tönen, die außerhalb des spielbaren Bereichs liegen.
- Sie befinden sich noch gar nicht im Laufzeitmodul. Nur das Laufzeitmodul erzeugt hörbare Töne, wenn Sie in der normalen Benutzerumgebung sind und auf der Klaviatur spielen, so wird keine Note an den Synthesizer geschickt!
- Sie haben, bevor Sie das Laufzeitmodul aufgerufen haben, sehr viele MIDI-Daten vom Masterkeyboard an den Computer geschickt (durch wildes Anschlagen von Tasten), so dass der MIDI-Puffer des Computers überlastet und das System abgestürzt ist. Dieser Effekt taucht zwar nur selten auf, trotzdem ist es sinnvoll, nur dann etwas auf der Klaviatur zu spielen, wenn das Laufzeitmodul auch aktiviert wurde und somit in der Lage ist, Ihre Noten auch auszuwerten.
- Eine Hardwarekomponente ist defekt. Z. B. ein MIDI-Kabel oder gar der Synthesizer oder das Masterkeyboard.

Wenn gar nichts hilft, so besteht die letzte Rettung darin, den Computer und alle anderen Komponenten auszuschalten und erneut zu starten, und zwar in der Reihenfolge Computer – Synthesizer. Sollten Sie auch dann noch kein positives Ergebnis erzielen, so sollten Sie mit uns Kontakt aufnehmen, um den Fehler zu beheben.

C.3.2. Die Intonation ist verkehrt

Dies liegt meistens daran, dass die Pitch-Bend-Range entweder bei allen oder bei einem MIDI-Kanal nicht korrekt eingestellt ist. Jede Klangfarbe muss eine Pitch-Bend-Range von 1 Halbton haben. Da die Pitch-Bend-Range meistens ein Parameter der eingestellten Klangfarbe und nicht ein globaler Parameter des Synthesizers ist, müssen Sie die Klangfarbe entsprechend modifizieren. Diese Einstellung ist nur in den seltensten Fällen kompliziert und ist dem Bedienungshandbuch des benutzten Synthesizers zu entnehmen. Informationen über Sinn und Zweck der Pitch-Bend-Range entnehmen Sie dem Abschnitt „Arbeitsweise der Synthesizertreiber“.

C.3.3. Es sind immer zwei Töne zu hören

Dieser Effekt tritt immer dann auf, wenn Klaviatur und Klangerzeugung in demselben Gerät stecken. Der eine Ton kommt vom Synthesizer selbst und intoniert die gewöhnliche gleichstufige Stimmung, der andere Ton ist der von Mutabor produzierte Mikroton. Um nur die Töne von Mutabor zu hören, müssen Sie Ihren Synthesizer in die MIDI-Betriebsart „LOCAL OFF“ versetzen, damit nur Töne erzeugt werden, die über MIDI geschickt wurden. Wie Sie diese Einstellung vornehmen, die bei einigen Synthesizern nach jedem neuen Anschalten des Gerätes vorgenommen werden muss, kann hier leider nicht allgemein gültig angegeben werden. Schlagen Sie hierzu im Bedienungshandbuch des Synthesizers nach.

C.3.4. Das Instrument ist nur monophon spielbar

Sie haben vergessen, den Synthesizer in den Multi-Mode zu versetzen und auf allen sechzehn MIDI-Kanälen dieselbe Klangfarbe einzustellen. Wie Sie Ihren Synthesizer richtig einstellen, lesen Sie auf Seite 90 und im Bedienungshandbuch des Synthesizers.

C.3.5. Es bleiben ungewollt Töne liegen

Dies ist ein Effekt, der eigentlich nicht auftreten sollte. Er kommt normalerweise nur bei einer Überlastung des Systems mit angeschlagenen Tasten (sehr viele Anschläge in extrem kurzen Zeiträumen) vor. Drücken Sie entweder die Panic-Funktion oder schalten Sie den Synthesizer aus- und wieder an oder wechseln Sie kurz die Klangfarbe und stellen Sie sie wieder her, da bei diesem Vorgang die meisten Synthesizer alle liegenden Töne ausschalten. Nur im größten Notfall ist es nötig, das Laufzeitmodul zu verlassen und erneut aufzurufen.

D. Beschreibung der Demonstrationslogiken „demo.mut“

Es ist von größter Wichtigkeit, dass Sie vor dem Aufruf der Demonstration alle Geräte angeschaltet, richtig miteinander verbunden und korrekt eingestellt haben. (*MULTI-MODE mit gleicher KLANGFARBE auf allen 16 Kanälen, PITCH-BEND-RANGE überall auf 1, MASTERKEYBOARD sendet auf KANAL 1* !!! Falls Sie Probleme mit der Einstellung haben, so lesen Sie sicherheitshalber die Abschnitte „Fehlerursachen“ und „Arbeitsweise der Synthesizer-Treiber“. Wenn Sie das Demonstrationsprogramm starten und alles funktioniert, so wird die Stimmung des Instruments auf eine gleichstufige Halbton-Temperierung eingestellt – also die ganz gewöhnliche Stimmung heutiger Instrumente.

D.1. G – Gleichstufige Temperaturen

Im Mittelpunkt der Stimmungslogik „GLEICH“ stehen gleichstufige Teilungen der Oktave[1]. Diese Stimmungslogik können Sie aktivieren, indem Sie die Taste 'G' drücken.

Innerhalb dieser Stimmungslogik stehen Ihnen fünf verschieden abgestufte gleichstufige Tonsysteme zur Verfügung: Halbtöne (12-Teilung der Oktave), Dritteltöne (18-Teilung der Oktave), Vierteltöne (24-Teilung der Oktave), Achteltöne (48-Teilung der Oktave) und Sechzehnteltöne (96-Teilung der Oktave). Außerdem können Sie auf den schwarzen Tasten eine Pentatonik (5-Teilung der Oktave) spielen.

D.2. H – Historische Stimmungen

Hier haben wir für Sie verschiedene historische Stimmungen von Musikinstrumenten (meist Klavier oder Orgel) vorbereitet[2]. Sie können diese sehr leicht mit der heute üblichen gleichstufigen Temperierung vergleichen, indem Sie einfach – auch bei liegenden Noten – zwischen der Stimmungslogik GLEICH und einem Tonsystem der Stimmungslogik HISTORISCH hin- und her schalten. Der Ton c hat in allen Tonsystemen die gleiche Frequenz. Bitte beachten Sie, dass das Aktivieren der Logik HISTORISCH noch *kein* neues Tonsystem einstellt. Wenn Sie also gerade eine gleichstufige Stimmung eingestellt haben und dann die Logik HISTORISCH durch Drücken der Taste 'H' aktivieren, so werden die Töne noch nicht verändert. Erst

[1] Frequenzfaktor $\sqrt[n]{2}$

[2] Den Stimmungen liegen die Schriften „Musikalische Temperierungen“ von Müller Schulze und „Das Stimmen“ von Heribert Schmitt zugrunde.

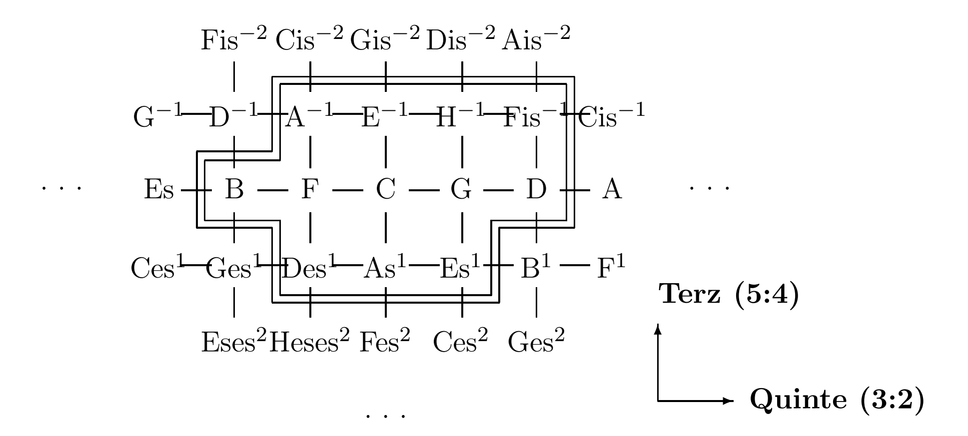
wenn Sie die Taste für eine konkrete Stimmung (innerhalb der Logik HISTORISCH) drücken, so wird diese eingestimmt.

Besonders bei diesen Stimmungen ist die Benutzung der Protokollfunktionen zum Anzeigen der aktuellen Stimmung sehr nützlich, um die Intervallstruktur der Tonsysteme zu erforschen (siehe Begleithandbuch „Die Benutzeroberfläche“.

Folgende historische Stimmungen sind programmiert: Pythagoreïsche Tonleiter, mitteltönige Stimmung, Silbermann, Werckmeister, Schlick, Kirnberger 3, „Bach“-Stimmung.

D.3. N – Tonales Netz

Diese mutierende Stimmung ermöglicht es, in immer reinen Dur– bzw. Mollakkorden zu spielen. Der Computer erkennt die Akkorde und intoniert Terzen und Quinten rein[3]. Das dieser mutierenden Stimmung zugrunde liegende Prinzip ist das „Wandern im tonalen Netz des (2,3,5)-frei erzeugten Tonsystems“. Man hört jeweils einen rhombusartigen Ausschnitt aus dem zweidimensionale Tonnetz (nach rechts sind reine Quinten, nach oben reine Terzen aufgetragen), wobei der aktuelle harmonische Grundton im Zentrum des Rhombus steht:



Das System wird auf C-Dur initialisiert. Es werden immer genau die Töne gespielt, die innerhalb des Rhombus liegen. Wird nun eine andere Tonalität gespielt, z. B. ein D-Dur-Dreiklang, so verschiebt sich der Rhombus, so dass sein Mittelpunkt über dem Ton D liegt. Dies bedeutet, dass gewisse Töne umgestimmt werden müssen, nämlich alle, die nicht in der Schnittmenge beider Rhomben liegen. Das wiederholte Spielen gewisser Kadenzen, z. B. C–d–F–G–C bewirkt somit eine immer weitere Entfernung vom Ausgangspunkt, was an einem deutlich hörbaren „Absinken“ der Stimmung zu bemerken ist (immerhin etwa einen Viertelton pro Kadenz).

[3] Dies entspricht einem auf „Modulation“ geschalteten einfachen Automaten zur Intonation in reiner Stimmung nach M. Vogel.

D.4. O - Obertöne

Eine sehr interessante Hörerfahrung bietet das Spielen mit der Obertonreihe. Die weißen Tasten dieses statischen Tonsystems sind aus einer Obertonreihe auf der Frequenz 110Hz aufgebaut, welche auf dem eingestrichenen c' beginnt und sich über drei Oktavlagen erstreckt. Auf den schwarzen Tasten liegt eine Untertonreihe auf demselben Grundton, allerdings um einige Oktaven nach oben verschoben. Probieren Sie einfach mal diese „Naturklänge“ aus - z. B. indem Sie Klänge aus geradzahligen mit ungeradzahligen Obertönen vergleichen, oder einen Cluster auf hohen schwarzen Tasten spielen, gefolgt von einem Cluster weißer Tasten ab dem c'. Das einfache Prinzip der Obertonreihe birgt zahllose neue Tonhöhenerlebnisse in sich ...

D.5. Modifizieren der Demo-Logiken

Die Demonstrationslogiken können Sie sehr einfach verändern und somit Ihre eigene Demonstration programmieren, da wir den Quelltext der Demonstration im Mutabor-Verzeichnis unter dem Namen `demo.mut` gespeichert haben.

Wenn Sie die Demonstrationslogiken verändern ist dieser Abschnitt des Handbuchs natürlich nicht mehr gültig. Er bezieht sich ausdrücklich auf die von uns ausgelieferte Version von `demo.mut`. Es könnte sehr lehrreich sein, wenn Sie sich einmal das Logikprogramm `demo.mut` im Editor ansehen würden, auch ohne es zu verändern, da in diesem Programm viele Elemente der Programmiersprache von Mutabor benutzt werden, die abzuschauen sich lohnen könnte.

E. Beispiele auf Diskette

Im Lieferumfang von MUTABOR sind verschiedene Beispielprogramme enthalten. Viele Programmierbeispiele aus dem Handbuch finden Sie hier bereits fertig getippt. Wenn dies der Fall ist, so ist in einer Fußnote der Programmname der Beispiellogik auf Diskette angegeben.

Um die Stelle im Handbuch zu finden, an der die Beispiele auf Diskette näher erläutert sind, finden Sie hier ein Verzeichnis der Beispiele auf der Beispieldiskette, sofern die Logikprogramme im Handbuch erwähnt werden.

Das Logikprogramm ...	finden Sie auf Seite ...
anker.mut	40
aufab.mut	71
auswahl.mut	81
c_b.mut	56
c_dur.mut	33
chaos.mut	88
drittel.mut	20
gleichsh.mut	33
komentar.mut	99
mininetz.mut	74
naturspt.mut	51
netz.mut	75
penta.mut	34
periode.mut	39
pythago.mut	65
xantippe.mut	73

F. Glossar

Grundparameter Die vier Grundparameter, die ein Tonsystem vollständig beschreiben sind Verankerungstaste, Breite, Töne der Fundamentaltonleiter und das Periodenintervall.

Fundamentaltonleiter Dieser im Handbuch vielleicht am häufigsten erwähnte Begriff bezeichnet einen bestimmten Blockbereich der Klaviatur, beginnend bei der Verankerungstaste, welcher genauso viele Tasten umfasst, wie die FT breit ist. In der Fundamentaltonleiter sind alle Informationen enthalten, die nötig sind, um den gesamten Klaviaturtasten Frequenzen zuweisen zu können.

Laufzeitmodul Der Teil von MUTABOR, welcher für das Live-Musizieren zuständig ist. Der Compiler generiert aus einem Logikprogramm ein eigenständiges Maschinenprogramm, das „Laufzeitmodul".

MIDI-Kanal Vergleichbar mit einem Radio kann der empfangende Synthesizer aus allen ankommenden MIDI-Meldungen die für ihn bestimmten herausfiltern. Wird z. B. über die MIDI-Leitung eine NOTE-ON-Meldung auf Kanal 5 gesendet, so reagieren nur diejenigen Geräte in der MIDI-Kette darauf, die als Empfangskanal ebenfalls den Kanal 5 eingestellt haben.

MIDI-Kette Werden mehrere MIDI-Geräte so miteinander verbunden, dass die Meldungen von Gerät A nach Gerät B nach Gerät C . . . geschickt werden, so spricht man von einer MIDI-Kette.

MIDI-Protokoll Genaue Spezifikation der Datenübertragung über die MIDI-Schnittstelle, z. B. Festlegung, dass Statusbytes das achte Bit gesetzt haben, dass #9x eine Note-On-Meldung ist, . . .

MIDI-Schnittstelle Genormte Verbindung zwischen Computern und Synthesizern bzw. Samplern. über diese Schnittstelle kommunizieren die Geräte und teilen sich z. B. mit, welche Tasten auf der Klaviatur gerade gedrückt wurden, welche Klangfarbe auf dem Synthesizer eingestellt werden soll, und vieles mehr.

Periodenintervall Das Periodenintervall bildet die Töne der Fundamentaltonleiter auf die restliche Klaviatur ab. Zwei Tasten, die im gleichen Abstand zueinander stehen, wie die Fundamentaltonleiter breit ist (in Halbtonschritten gerechnet), stehen zueinander im Frequenzverhältnis des Periodenintervalls. (siehe Seite 28)

Pitch-Bender Die meisten Masterkeyboards haben einen Dreh-Regler, mit dem man die Tonhöhe aller liegenden Töne quasi stufenlos erhöhen bzw. erniedrigen kann. Spaltet man den Synthesizer intern in bis zu 16 eigenständige Instrumente mit derselben Klangfarbe auf, so kann jeder Kanal einen eigenen Pitch-Bend-Wert, also eine eigene Verstimmung bekommen. Auf diese Weise entlockt MUTABOR nahezu jedem handelsüblichen Synthesizer/Sampler die Mikrotöne.

Sequenzer Ein Computerprogramm, das wie ein Mehrspur-Aufnahmegerät auf dem MIDI-Keyboard gespielte Noten aufzeichnen, bearbeiten, speichern und wiedergeben kann.

Töne der Fundamentaltonleiter Die Töne der Fundamentaltonleiter enthalten die Information, welche Frequenzen den Tasten der FT zugeordnet sind, bzw. welche Tasten gesperrt sind.

Verankerungstaste Die unterste Taste der Fundamentaltonleiter wird Verankerungstaste genannt. Sie muss nicht notwendigerweise einen Ton tragen, kann also durchaus gesperrt sein.

G. Schlussbemerkung

Wir haben versucht, die Programmiersprache zum Erstellen von Tonsystemen und Stimmungslogiken klar und verständlich zu gestalten, so dass die zugrunde liegenden Konzepte prägnant zum Ausdruck kommen.

Der Leser bzw. die Leserin sei an dieser Stelle ermutigt, sich durch praktische Anwendung mit MUTABOR vertraut zu machen. Die Programmiersprache MUTABOR ist eine Art Universalinstrument, von dem wir selbst nicht vorhersehen können, was damit alles möglich ist. Demzufolge hat MUTABOR auch den Charakter eines Experimentalinstruments. Es ist nun die Sache des Musikers und Komponisten, die verschiedenen Möglichkeiten dieses Instruments zu erforschen.

Wir selbst betrachten MUTABOR nicht als endgültig feststehende Programmiersprache, sondern als Experimentiersprache, die ihrerseits weiter entwickelbar ist. Wohin diese Entwicklung gehen soll und wird, hängt maßgeblich von der Resonanz aus der Musikwelt ab. Für Lob und Kritik, insbesondere aber für Verbesserungsvorschläge haben wir stets ein offenes Ohr.

Wir wünschen Ihnen auf jeden Fall viele neue musikalische Erfahrungen mit MUTABOR.

Index